Université de Montréal

Extraction des patrons de régime de VerbNet pour une implémentation dans un système de génération automatique de texte

par

Daniel Galarreta-Piquette

Département de traduction et linguistique
Faculté des arts et des sciences

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

22 février 2018

SOMMAIRE

Ce mémoire explore les patrons de régime en anglais provenant de la ressource lexicographique *VerbNet* et leur implémentation dans un système de génération automatique de texte.

SUMMARY

English summary and keywords...

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET DES ABRÉVIATIONS

GAT	Génération automatique de texte
GP	Patron de régime, de l'anglais *Government Pattern*
DPOS	Partie du discours profonde, de l'anglais *Deep Part of Speech*
TST	Théorie Sens-Texte
VN	*VerbNet*

DÉDICACES

Vos dédicaces.

REMERCIEMENTS

Remerciements...

INTRODUCTION

Parler de la génération de texte automatique, de *VerbNet*, de la problématique (pas de consensus quant à l'architecture de la classe verbale en TAL) de VerbNet, des dictionnaires. Pourquoi les verbes sont si importants ? (Kipper,2005, dissertation) Puisque les verbes sont porteurs du sens principal de la phrase, il faudrait donc créer une ressource qui puisse rendre compte du sens des verbes si on souhaite une bon fonctionnement des applications NLP.

Chapitre 1

EXTRACTION DE L'ARCHITECTURE DE *VERBNET*

Dans ce chapitre, nous verrons l'apport que la ressource lexicale *VerbNet* peut offrir à des applications en traitement automatique du langage (TAL). Nous avons comme objectif d'extraire l'architecture de dictionnaire *VerbNet* pour l'implémenter dans un dictionnaire de type sens-texte qui servira à générer du texte. Cet objectif provient d'une problématique que nous avions rencontré auparavant. Nous voulions savoir comment organiser notre dictionnaire en ce qui concernait les verbes. Car ceux-ci sont si riches et complexes qu'il nous fallait trouver un moyen systématique d'encoder cette partie du discours. De plus, il nous fallait l'encoder pour que notre nouveau dictionnaire puisse intéragir correctement avec les règles grammaticales du logiciel. En ce qui concerne les dictionnaires, parmi la communauté TAL, il ne semble pas y avoir de consensus quant à la manière de procéder pour modéliser la classe verbale. La raison est simple, les verbes démontrent des comportements variables, très riches au niveau de l'éventail de patrons de régime possibles pour un même verbe, et assez complexes ce qui nécessite beaucoup plus d'attention que d'autres parties du discours comme les noms qui démontrent beaucoup moins de variétés d'usage quant au nombre de patrons de régime. Ce qui fait en sorte que comme tous les verbes sont des prédicats, et que les prédicats sont les noyaux des énoncés, il faut les traiter avec soin, faute de quoi leur application en NLP sera médiocre. Cette problématique nous a donc amené à constater que VerbNet s'était penché sur ce problème et nous voulions vérifier si rouages de leur dictionnaire peuvent s'appliquer en génération automatique de texte. De nos jours, avec les modèles stochastiques où il n'y a pas d'analyse linguistique, on ne construit pas de dictionnaires ou de règles grammaticales, mais on le laisse le soin au système d'apprendre les règles par lui-même et de développer le lexique par lui-même en n'assignant pas de POS et en voyant les langues naturelles comme des suites de caractères. C'est une mode qui fonctionne présentement grâce à la quantité immense d'information qu'on retrouve et ces nouveaux corpus

incroyables combinés avec la puissance des ordinateurs et l'apprentissage machine, certains font complètement fi de l'analyse linguistique dans leurs applications TAL. Toutefois, les utilités principales de ces systèmes sont de divers ordres de NLP, mais en en ce qui concerne la génération automatique de texte, le traitement de la langue se doit d'être impeccable. D'où la nécessité de développer des outils puissant et rigoureux qui nécessiteront plus de temps à développer car il faudra créer les règles et enrichir le dictionnaire qui compose la langue. Pour ce faire, on doit recourir à des méthodes linguistiques qui étaient plus populaires dans la fin des années 90 et début 2000 (citer des systèmes). Toutefois, en étant bien conscient du changement de cap, nous pensons qu'il est encore primordial de développer de bons outils linguistiques computationnels car c'est de cette manière qu'on pourra le mieux représenter le fonctionnement de la langue. De plus, nombres de chercheurs combinent l'apprentissage machine à des règles et des dictionnaires pour ainsi développer le lexique partiellement en annotant automatiquement des lexèmes.(trouver une citation dans les INLG de 2014 à 2017)

1.1. VerbNet

Ainsi, tel que mentionné précédemment, VerbNet a été créé dans un contexte où il y avait un réel besoin de réfléchir à la meilleure manière de procéder pour construire un dictionnaire qui saura tenir compte de la richesse et la complexité que renferment les verbes (Kipper, Dang et Palmer, 2000). C'était en réponse au manque de lignes directrices sur l'organisation des verbes dans les dictionnaires destinés à des applications TAL et ce malgré la quantité impressionnante de dictionnaire computationnels existants déjà. Parmi ceux qui ont tenté la chose, nous nommerons : le generative lexicon de Pustojevsky (Pustejovsky, 1991) WordNet de Fillmore (Miller, 1995), ComLex(Grishman, Macleod et Meyers, 1994), LCS(Ayan et Dorr, 2002) et FrameNet(Baker, Fillmore et Lowe, 1998). Ceux-ci comportaient des lacunes que du point de vue du traitement des entrées lexicales verbales, selon les auteurs de VerbNet. Les lacunes étaient diverses mais importantes, par exemple, ils trouvaient que le Generative Lexicon ne focusait que sur les noms, que WordNet ne donnait pas de détails concernant les cadres syntaxiques possibles et aucune mention de la structure prédicat-argument. Quant à ComLex, ils décrivaient effectivement les cadres syntaxiques possibles, mais ne faisait pas de désambiguisation sémantique. Finalement, ils jugeaient que le lexicon LCS a essayé de palier à ces lacunes aussi, mais l'étentue lexicale qu'il offrait n'était pas assez large à leur goût(Schuler, 2005). C'est donc dans ce contexte qu'est né le projet VerbNet.

1.1.1. Levin et l'organisation en classes verbales

Le travail de Levin était de créer un lexicon verbal de l'anglais. Elle a fait des recherches sur les propriétés sémantiques et syntaxiques des verbes. Son travail a été fait avec l'optique que le comportement que les verbes démontrent par rapport aux arguments qu'ils sélectionnent est largement déterminé par les composantes sémantiques des verbes. Elle a essayé de délimiter et systématiser toutes les facettes des divers comportements des verbes. Comme les verbes sont tous des prédicats et qu'ils forment le noyau des énoncés, Levin s'est intéressée à eux. De plus, les verbes démontrent des propriétés très complexes contrairement aux autres parties du discours. Il existe un éventail de combinaisons possibles pour un verbe avec ses arguments. Tout locuteur natif d'une langue sait quels sont les diathèses possibles et les arguments possibles d'un verbe et même parfois pour des subtilités de la langue que la plupart ne pourrait expliquer.

Son exemple de base était le suivant. Prenons des locuteurs natifs, ils savent que des verbes comme Spray et Load permettent 2 différentes alternances (écrire l'exemple) Puis, elle veut ainsi nous amener à voir que le partage de ces alternances de diathèses entre groupe de verbes démontrent qu'ils partagent aussi des caractéritiques sémantiques similaire et que donc, ces verbes devraient être regroupés en classe. Par la suite, elle présente dans son livre, toutes les sortes de classes possibles de la langue anglaise. Elle choisit un verbe représentatif de la classe pour la décrire, puis une liste de membres qui peuvent se combiner de la même manière.

C'est en regroupant les verbes en fonction de leur comportement (la manière qu'ils se combinent, leur sous-catégorisation) qu'on remarque qu'il semble y avoir des classes verbales. Autrement dit, qu'il y a des verbes qui se ressemblent assez qu'on pourrait les regrouper ensemble, ce qui nous amène à vouloir classer systématiquement tous les verbes d'une langue à l'intérieur d'une de ces classes construites.Tel que Levin le dit (Levin, 1993), son travail a été guidé par l'idée que les comportements syntaxiques des verbes sont motivés par les composantes sémantiques qui les représente. À ces fins, Levin a observé les comportements des verbes en anglais pour y tester cette hypothèse linguistique car il pensait que c'était une bonne manière de démontrer la chose. Si des verbes qui partagent le même type de comportement syntaxique, ils doivent aussi probablement partager des caractéristiques sémantiques. Son travail a été de classer les verbes systématiquement. Déjà , Levin soulignait l'importance d'encadrer les verbes dans les lexicons car les verbes qui sélectionne des arguments démontrent un ensemble complexes de propriétés. Ainsi, au lieu de construire un dictionnaire où chaque entrée verbale est prise individuellement, on regroupe les entrées lexicales en classe en fonction de leur comportement syntaxique et sémantique.

À la base le traitement est fait en fonction du comportement syntaxique, puis selon Levin, les verbes qui partagent le même type d'alternations syntaxiques (patrons de régime) partagent aussi la sémantique sous-jacente à ces constructions, donc cela fait en sorte qu'ils partagent généralement les mêmes caractéristiques sémantiques aussi. Le fait d'organiser le dictionnaire en classe verbale accélère le processus de création du dictionnaire, car une fois qu'on détermine le nombre de verbes qui fonctionnent de la même manière, il ne reste plus qu'à faire une entrée qui retient l'information syntaxique et sémantique partagée par l'ensemble de la classe, et passer à la classe suivante. Après, lorsqu'on a fait le tour des verbes, on a juste à les ajouter à l'une des classes pré-existantes. Les créateurs de VerbNet sont conscients du fait qu'on va parfois tenter de faire rentrer un verbe dans une classe et que c'est un brin tirer par les cheveux, mais il s'agit d'une infime partie du dictionnaire. (Schuler, 2005). Ainsi, les verbes d'une même classes partagent des comportements syntaxiques similaires. Ce qui a été démontré par Jackendoff, qui montre qu'à travers les langues, on peut identifier des classes verbales. Plusieurs applications TAL ont tiré profit d'organiser les verbes en classes verbales (Schuler, 2005), mais contrairement à ce que VerbNet et nous essayons de faire, ces dictionnaires étaient domaine-spécifique et non domaine-indépendant.

Les créateurs de VerbNet ont jugé qu'il serait nécessaire d'organiser leur information en classes verbales. Ce qui est un premier pas vers une compréhension de leur hypothèse sur la manière d'organiser un dictionnaire autour de verbes. Ce que nous avons fait par le fait même puisque c'est cette idée que nous avons recyclé de VerbNet. Ça s'est avéré effectivement tel que VerbNet l'a explicité. L'idée d'organiser l'information ainsi leur est venu de Levin qui avait déjà commencé classer les verbes et avait fait un travail considérable sur ce sujet.

Comment fonctionnent les classes de Levin ? Elles démontrent très clairement le comportement syntaxique des verbes, mais ne tiennent pas compte des composantes sémantiques d'une classe de verbe. Les classes regroupent des verbes qui agissent de la même manière d'un point de vue syntaxique. Autrement dit, des verbes qui participent aux mêmes cadres syntaxiques sans que ça n'altère leur sens. De plus l'aspect sémantique de la chose vient du fait que Levin soutenait que sans que les verbes veuillent dire la même chose, les verbes avec une sémantique similaire démontreront aussi des caractéristiques syntaxique similaires. (d'un point de vue de la structure argumentale). Les cadres syntaxiques associés aux classes sont supposés représentés les composantes sémantiques sous-jacentes et communes à une classe de verbe. Ainsi, en fonction du sens d'un verbe, dans les aspects primitifs des caractéristiques sémantiques, certains verbes vont agir de la même manière car ils partagent cette caractéristique sémantique. Certains verbes partargeront ainsi une partie des cadres syntaxiques, mais pas tout, car une composante sémantique leur appartenant fera en sorte qu'un autre

cadre syntaxique serait incorrect. Voir l'exemple suivant, tiré de (Schuler, 2005) et à la base de (Levin, 1993).

Ici, on voit que les verbes spray/load peuvent exprimer leurs arguments d'au moins deux manières différentes. La seconde manière que Levin a appelé "locative alternation".

Voici la mentalité de Levin en action :

(1) Cadre syntaxique : Spray

a. Sharon sprayed water on the plants.
b. Sharon sprayed the plants with water.

(2) Cadre syntaxique : Load

a. The farmer loaded apples into the cart.
b. The farmer loaded the cart with apples.

Ces mêmes locuteurs natifs savent que des verbes qui s'apparentent à Spray/Load (car ils semblent correspondre au même type d'évènement) ne permettent pas les 2 alternances démontrées en (1) et en (2). Les verbes fill/cover permettent l'option b) et les verbes dump/pour permettent l'option a). Ainsi, à partir de tests de ce type, pour voir quels verbes permettaient le même type de construction, elle a bâti un lexicon qui regroupe les verbes en fonctions des alternances de diathèses qu'ils permettent.

(3) Alternances de diathèses : cover

a. *Monica covered a blanked over the baby.
b. Monica covered the baby with a blanket.

(4) Alternances de diathèses : fill

a. *Gina filled lemonade into the pitcher.
b. Gina filled the pitcher with lemonade.

(5) alternances de diathèses : pour

a. Carla poured lemonade into the pitcher.
b. *Carla poured the pitcher with lemonade

(6) alternances de diathèses : dump

a. The farmer dumped apples into the cart.
b. *The farmer dumped the cart with apples.

Ainsi, en opposant diverses diathèses possibles, Levin a fait un travail collossal où elle a repertorié des verbes se combinant de la même manière. C'est pourquoi des verbes qui ont beau être des synonymes, ne se combinent pas de la même manière. Elle avait ainsi un éventail de diathèses à tester et elle regroupait en classe les verbes qui partagaient les mêmes alternances. Se justifiant ainsi en disant que les raisons qui sous-tendent un partage des diathèses proviennent d'une sémantique similaire.

(cette partie manque l'exemple venant de verbnet)Ainsi, tel que démontré en **??** et **??** les verbes appartenant à la classe break et à la classe cut se ressemblent car ils peuvent tous les 2 prendre ces cadres syntaxiques (ce genre de construction syntaxique). Toutefois, en **??** et en **??**, on voit très bien qu'il ne partagent pas ces constructions syntaxiques. *Break* prend seulement la construction intransitive et exclut l'autre, tandis que *cut* prend la construction conative et exclut l'intransitive. La raison que Levin donne est la suivante. Le verbe *cut* décrit une série d'actions ciblant la complétion d'un but (séparer un objet en morceau). Toutefois, il est possible de faire ces séries d'actions sans que l'objectif final ne soit atteint, mais l'action de couper peut quand même être perçue. En ce qui concerne *break*, la seule chose qui importe dans l'évènement, c'est le changement d'état d'un objet (qui devient séparés en morceaux). Si on n'arrive pas au résultat, une tentavive de briser quelque chose ne peut être perçue. Ce qu'on peut tirer de cet exemple, est que les classes verbales regroupent des verbes qui partagent des comportements syntaxiques similaires, les membres des classes ne sont donc pas nécessairement des synonymes, il ne s'agit que de verbes qui s'utilisent de la même façon. Ainsi, certaines classes vont effectivement regrouper des verbes qui signifient à peu près la même chose, mais aussi des verbes qui en surface, ne partagent pas bcp avec une majorité des verbes de cette classe, mais syntaxiquement, ces membres se comportent comme le reste de la classe. Ainsi, selon Levin, il y a quelque chose derrière les composantes sémantiques de ces membres qui les unirait avec les autres qui se ressemblent.

Les auteurs de VerbNet ont aussi ajouter des classes à ce que Levin avait fait. Ce qu'ils appellent intersective Levin classes, ce sont des sous-ensembles de classes qui s'entrecoupent et forment des classes à part. (Schuler, 2005) et (Kipper, Korhonen, Ryant et Palmer, 2006).

De plus, les structures de Moens et Steedman (Moens et Steedman, 1988), basés sur les travaux de Vendler confirment l'hypothèse de Levin. La sémantique des évènements qui relie des verbes entre eux, correspond sans problème aux verbes qui se retrouvent dans les classes de Levin (ce qui est confirmé par le postulat de Levin qui pense que le fait que certains verbes agissent de la même manière est due au fait qu'ils partagent une sémantique sous-jacente similaire bien qu'il ne soient pas des synonymes).

1.1.2. Composantes de VerbNet

VerbNet est composé de classes verbales, tel que nous l'avons mentionné précédemment, et pour les raisons évoqueés. Chaque classe contient un ensemble de membres qui lui sont attribués en fonction du caractère commun que les membres possèdent avec la classe, des rôles thématiques pour la structure argument-prédicat et des restrictions sélectionnelles sur ces arguments ainsi que les cadres syntaxiques qui contiennent une courte description du cadre, un exemple le démontrant et une description syntaxique du cadre. Puis finalement, un ensemble des prédicats sémantiques. Une classe peut être divisé en sous-classe si cela est nécessaire. Tel sera le cas lorsque un sous-ensemble des membres de la classe partage des cadres syntaxiques et des prédicats sémantiques spécifiques. Ce qui est l'aspect hiérarchique de la chose.

1.1.2.1. *Organisation hiérarchique*

Hérité de Levin (Levin, 1993)

Les raisons qui ont motivé VerbNet a organisé son information en hiérarchie. D'abord, il se sont fortement inspiré de Acquilex Lexical Knowledge Base (trouvé la référence exacte). Ensuite, pour des raisons pratiques, ils ont trouvé que c'était la meilleure manière d'organiser cette montagne d'information lexicale car ils ont revu l'information fournie par Levin et ont ajouté des sous-classes aux classes ainsi que des sous-sous-classes aux sous-classes, allant jusqu'à un niveau de 3 steps de profondeur. Donc, des sous-ensembles des sous-ensembles. Ce qu'ils considèrent comme une raffinement des classes originales de Levin (Schuler, 2005). En gros, c'est l'ajout de sous-classes aux classes origniales de Levin (et aux sous-sous-classes) qui ont rendu VerbNet hiérarchique.

Le fonctionnement est assez simple. Une sous-classe fille hérite de toute l'information de sa classe mère. Les sous-classes sont à l'origine de vouloir spécifier le comportement qui rassemble un sous-ensemble des membres d'une classe. Ainsi, une sous-classe est un ajout d'information par rapport à des restrictions d'usages de rôles thématiques, de cadres syntaxiques ou de prédicats sémantiques.(guidelines, VerbNet : ....com)

Prenons un exemple, la classe verbale *Transfer of a Message* séparer l'exemple en 3 parties (en fonction des niveaux).

Cette classe comporte trois niveaux, en termes de hiérarchie. Au premier niveau, nous avons les rôles thématiques, cadres syntaxiques et prédicats sémantiques partagés par tous les membres de la classe. En ce qui concerne cette classe, il y a au premier niveau, la spécificité qu'il s'agit d'une classe verbale mère avec son ID et VNCLASS. Puis les membres de la classe.

Puis les rôles thématiques. Puis la section FRAMES qui contient tous les FRAME. Chaque FRAME est composé de : une description, un exemple, la syntaxe, puis la sémantique. Ainsi, au premier niveau, on retrouve 10 FRAME dans la section FRAMES. Par après, on accède au second niveau, qui est celui des sous-classes. Il n'existe qu'une sous-classe dans cet exemple, mais certaines classes verbales de VN contiennent plus d'une sous-classe au même niveau. Lorsque c'est le cas, il s'agit de classes soeurs. qui sont toutes deux enfant du même parent. Revenons à notre cas, ainsi, nous n'avons qu'une sous-classe au second niveau. Cette sous-classe garde la même structure que la classe mère. Ainsi, d'abord, on a la précision qu'il s'agit d'une sous-classe et son identification. Puis on a ses membres, suivi des rôles thématiques, dans ce cas-ci, il n'y a pas d'ajout à faire pour cette section puisque la sous-classe utilise les mêmes rôles thématiques que la classe mère, ainsi, par héritage, elle hérite de tous les rôles thématiques mentionnées précédemment. Puis on entre dans sa section de FRAMES qui contient tous les FRAMES appartenant à cette sous-classe. Ainsi, les membres de cette classe hérite des FRAMES de la classe mère en plus des FRAMES qui leurs sont plus spécifiques, car ces membres partagent des caractéristiques syntaxiques et sémantiques de plus que par rapport aux membres de la classe mère. Dans cet exemple, il y a un FRAME. Puis on accède au troisième niveau. Une sous-classe de la sous-classe précédente. Ainsi, le système qu'a adopté VerbNet pour distinguer les concepts de child/sister/parent sont des numérotation des classes (et sous-classes). Ainsi, on sait à coup sûr qu'il s'agit d'une sous-classe de la sous-classe car, au second niveau il s'agissait de la sous-classe transfer mesg-37.1.1-1 et au troisième niveau on a transfer mesg-37.1.1-1-1 et s'il y avait plus qu'une sous-classe à ce niveau, on aurait vu transfer mesg-37.1.1-1-2.

Les classes verbales de VerbNet sont numérotées par des chiffres allant de 9-109 (guidelines). Ainsi, le numéro apparaissant devant une classe verbale est associé à des caractéristiques sémantiques et syntaxiques communes. Par Exemple, les classes verbales associées à des verbes de type "mettre quelque chose" commenceront par le chiffre 9. Ce qui nous donne quelque chose comme :

— put 9.1
— put spatial 9.2
— funnel 9.3
— put direction 9.4
— pour 9.5
— coil 9.6
— spray 9.7
— fill 9.8
— butter 9.9

— pocket 9.10

Certains numéros n'impliquent qu'une seule classe, car il n'y a pas d'autres classes qui partagent ce genre de traits sémantiques ou syntaxiques communs.

Par la suite, on se sert encore de la numérotation pour expliciter la hiérarchie à l'intérieur même d'une classe (guidelines). Chaque classe peut inclure des classes filles, qui sont des classes sœurs entre elles, et qui peuvent avoir des classes filles à leur tour. La classe verbale Spray en démontre bien la chose.

D'abord, il y a la classe supérieure, qui est la plus haute de la hiérarchie, toutes les caractéristiques de cette classe sont partagées par tous les verbes de la classe. Dans la top classe nous avons les constructions syntaxiques et les prédicats sémantiques partagés par la classe, ainsi que les rôles thématiques.

Une classe mère domine une sous-classe, toutes ses caractéristiques sont partagées avec les classes subordonnées à celle-ci.

Une sous-classe : Les sous-classes dans VerbNet héritent des caractéristiques provenant de la classe dominante, mais elles spécifient particulièrement des constructions syntaxiques et une sémantique entre les verbes membres de cette sous-classe. Ce qui est spécifié dans une sous-classe peut être de différents ordres : ajouter des constructions syntaxiques propres à ce sous-groupe, ajouter des restrictions sélectionnelles sur les étiquettes des rôles sémantiques.

Des classes sœurs, ne partagent pas de caractéristiques hormis celle héritées par leur classe mère. Le reste de l'information encodée dans la classes sœurs n'est valide qu'à l'intérieur de leur classe respectivement.

Démontrer la chose comme dans Guidelines :

— Spray-9.7
— Spray-9.7-1
— Spray-9.7-1-1
— Spray-9.7-2

1.1.2.2. *Rôles thématiques*

VerbNet utilise un ensemble de 23 rôles thématiques pour identifier les arguments dans les classes verbales (vérifier de quels rôles thématiques ils s'inspirent).On étiquette les arguments dans les classes verbales en leur associant un rôle thématique. La raison pour laquelle VerbNet a opté pour les rôles thématiques est que, contrairement à un étiquetage générique où on énumère les arguments en procédant comme "Argument 1" Verbe "Argument 2" pour

illustrer un cadre syntaxique est parce que les rôles thématiques peuvent offrir de l'information sémantique de plus que juste un argument numéroté. Effectivement, à la base, les rôles thématiques ont été pensés dans les années 60 pour assigner une fonction aux arguments sélectionnés par des prédicats. On les avait créé pour typer les participants des prédicats. La spécification du rôle fournit de l'information sémantique sur le type d'argument en jeu, tandis que numéroté ne fournit rien du tout. Chaque argument se fait donné un rôle thématique unique, et ces rôles thématiques sont partagés par tous les membres d'une classe. Donc, ils doivent être assez large pour que ce soit cohérent avec tous les membres, mais pas trop imprécis non plus. Les rôles thématiques correspondent à la relation sémantique qui existe entre un prédicat et ses arguments.

VerbNet utilise les rôles thématiques pour étiquetter les arguments figurant des les cadres syntaxiques (et sémantiques) (Schuler, 2005). Elle puise dans une banque de 23 rôles thématiques pour associer le bon rôle à l'argument en question. Ils ont choisi ces rôles car ils étaient assez généraux pour se prêter à toutes les évènements que dégagent les verbes dans le dictionnaire. Ils voulaient capturer l'essence des verbes et démontrer encore une fois le caractère général des verbes en démontrant qu'une poignée d'argument peut bel et bien rendre compte des arguments sélectionnés par les verbes en anglais. À l'intérieur d'une même classe verbale on y retrouve un nombre x de rôles thématiques qui seront mappés aux arguments sélectionnés dans les cadres syntaxiques et sémantiques fournis par VerbNet. Ils ont choisi les rôles thématiques au lieu d'autres moyens d'étiquettage des arguments car ils trouvaient que ça ajoutait de l'information sémantique sur une classe verbale.

Les rôles thématiques choisis par VerbNet sont de divers ordres, certains proviennent des premiers pensés par (Fillmore,1968) d'autres de Jackendoff (Jackendoff, 1972), et des ajouts pour compenser pour les classes qui nécessiteraient des rôles plus spécifiques à leur classes, tout en étant assez généraux pour pouvoir se prêter à d'autres classes. Encore une fois, tel que mentionné dans le background de Dissertation, certains critiques les rôles thématiques car ils ont un caractère assez arbitraires et il n'est pas facile de tout justifier. De cette manière, les rôles thématiques utilisés par VerbNet n'ont pas été créé systématiquement, mais plus en fonction de ce qui cadrait le plus avec les classes verbales et les arguments nécessaires qui apporteraient le plus d'information sémantique pertinente. Ils ne pensent pas qu'un ensemble plus petits de rôles thématiques aurait pu être suffisant, mais ils ne pensent pas non plus que ceux qu'ils ont choisi sont exactement ce dont on aurait besoin, il n'ont pas nécessairment été au bout de tous les rôles thématiques possibles. Mais, ils considèrent que ce qu'ils avaient était suffisant pour rendre compte des comporetements des arguments envers leur prédicats respectifs, et ce pour toutes les classes verbales.

À l'intérieur d'une classe verbale (ou sous-classe, etc.), chaque argument se fait assigner un rôle thématique

Les rôles thématiques nous donne de l'information sur la sémantique des participants d'une phrase dans le but que les différents cadres syntaxiques (alternances) n'influencent pas les rôles assignés. On nous fourni un exemple que nous réutiliserons pour illustrer le propos (guidelines) :

(7) a. Sandy shattered the glass.
b. The glass shattered

D'un point de vue de la syntaxe, dans la première phrase, Sandy est le sujet du verbe et the glass en est l'objet direct. Tandis que dans la seconde phrase, the glass est le sujet du même verbe, et il n'y a pas d'objets directs. En leur assignant un rôle thématique, on notera que d'un point de vue de la sémantique rien n'a changé, même si la structure a changé en apparence. Ainsi, si on assigne un rôle d'Agent à Sandy et un rôle de Patient à the glass, on remarque que les rôles sémantiques sont cohérents avec la phrase, malgré le changement de sujet et d'objet direct.

Les rôles thématiques sont explicités au début d'une classe verbale. (exemples venant de break-45.1.xml)

Listing 1.1. les rôles thématiques

```
<THEMROLES>
    <THEMROLE type="Agent">
        <SELRESTRS>
            <SELRESTR Value="+" type="int_control"/>
        </SELRESTRS>
    </THEMROLE>
    <THEMROLE type="Patient">
        <SELRESTRS>
            <SELRESTR Value="+" type="solid"/>
        </SELRESTRS>
    </THEMROLE>
    <THEMROLE type="Instrument">
        <SELRESTRS>
            <SELRESTR Value="+" type="solid"/>
        </SELRESTRS>
    </THEMROLE>
    <THEMROLE type="Result">
            <SELRESTRS/>
    </THEMROLE>
</THEMROLES>
```

Finalement, les rôles thématiques sont domain-general, voulant dire qu'on veut les réutiliser dans le plus de classes là où c'est pertinent. Ils ne sont pas spécifiques à une classe ou un verbe, on les retrouve partout dans le lexicon. C'est pourquoi on veut qu'ils soient assez général pour rendre compte de divers type d'arguments.

Pour les besoins de notre travail, nous n'utilisons pas les rôles thématiques dans notre travail, mais nous voulions souligner qu'ils étaient importants pour les créateurs de VerbNet. Voir les raisons de Melcuk p.230

1.1.2.3. *Restrictions sélectionnelles*

Les restriction sélectionnelles vont sur les rôles thématiques. Est-ce que c'est pertinent d'en parler ? Probablement, pas, on ne s'en sert pas du tout. En gros, il s'agit de restrictions imposées aux rôles thématiques pour que ça sélectionne un certain type d'argument. Ainsi, parmi ces restrictions, parfois, certains verbes requièrent que l'Agent soit nécessairement de type x, tandis que le patient peut être n'importe quoi. OU l'inverse, etc. Ça donne encore plus d'informations sur le participant de la phrase, de plus, VerbNet considère que ces restrictions sélectionnelles offrent encore plus d'information sémantique, car on précise le type d'argument requis.

Listing 1.2. les restrictions sélectionnelles

```
<THEMROLES>
    <THEMROLE type="Agent">
        <SELRESTRS>
            <SELRESTR Value="+" type="int_control"/>
        </SELRESTRS>
    </THEMROLE>
    <THEMROLE type="Patient">
        <SELRESTRS>
            <SELRESTR Value="+" type="solid"/>
        </SELRESTRS>
    </THEMROLE>
    <THEMROLE type="Instrument">
        <SELRESTRS>
            <SELRESTR Value="+" type="solid"/>
        </SELRESTRS>
    </THEMROLE>
    <THEMROLE type="Result">
            <SELRESTRS/>
    </THEMROLE>
</THEMROLES>
```

1.1.2.4. *Cadres syntaxiques*

Pour une classe donnée, on y retrouve soit un ou des cadres syntaxiques qui servent à représenter le type de réalisation de surface possible pour cette classe verbale. D'ailleurs, ces cadres syntaxiques sont partagés par l'ensemble des membres d'une classe ou d'une sous-classe. Chaque cadre syntaxique décrit une construction verbale de type transitives directes/indirectes, des intransitives, des phrases prépositionnelles, etc. Un cadre syntaxique est composé de rôles thématiques dans leur position argumentale ainsi que le verbe qui les régie (ainsi que d'autres unités lexicales nécessaires au bon fonctionnement d'une construction).

À l'intérieur de chaque classe verbale, on retrouve la liste des membres, suivie de la liste des rôles thématiques qu'on retrouve dans les cadres syntaxiques de cette classe, puis les cadres syntaxiques. Ainsi, on liste tous les cadres syntaxiques possibles pour une classe. Ce qui nous donne de l'information de nature syntaxique et explicite les liens qui unissent les rôles thématiques au verbe en question. Ça nous donne aussi beaucoup d'information concernant le verbe, car on voit comment il peut être utilisé, quel genre d'actant il sélectionne, de quel type de patron de régime il s'agit. C'est ce qui nous intéresse le plus à la base. C'est de savoir comment ce verbe se combine, peut-il prendre un verbe comme complément d'objet direct, est-ce qu'il sélectionne telle ou telle préposition. Bref, l'information contenue dans les cadres syntaxiques est très pertinente pour notre travail, car il existe très peu de dictionnaires qui ont voulu énumérer toutes les alternances syntaxiques d'un verbe. Puis, les cadres syntaxiques respectent aussi la caractéristiques hiérarchique omniprésente dans VerbNet. Ainsi, tous les membres d'une classe partagent ces patrons de régime, et une sous-classe hérite aussi des patrons de régime de la classe qui la gouverne.

guidelines : Les cadres syntaxiques sont compris dans la section FRAMES de VerbNet qui contient, les cadres syntaxiques ainsi que les prédicats sémantiques. Cela nous donne une description des différentes réalisation en syntaxe de surface et des alternances de diathèses permises pour les verbes représentés par cette classe. On a ainsi une vue d'ensemble sur les constructions syntaxiques possibles par classe verbale.

Dans la figure ci-bas, on voit le début de FRAMES, qui est la balise comprenant tous les FRAME qui eux sont des cadres. À l'intérieur de FRAME, on a SYNTAX et SEMANTICS. Commençons d'abord par SYNTAX, et faisons abstraction de DESCRIPTION et EXAMPLES pour l'instant.

Listing 1.3. cadres syntaxiques

```
<FRAMES>
        <FRAME>
```

```
            <DESCRIPTION descriptionNumber="0.2" primary="NP V NP"
                         secondary="Basic Transitive; Causative" xtag="0.2"/>
            <EXAMPLES>
                <EXAMPLE>Tony broke the window.</EXAMPLE>
            </EXAMPLES>
            <SYNTAX>
                <NP value="Agent">
                    <SYNRESTRS/>
                </NP>
                <VERB/>
                <NP value="Patient">
                    <SYNRESTRS/>
                </NP>
            </SYNTAX>
```

1.1.2.5. *Prédicats sémantiques*

VerbNet est fier de pouvoir offrir de l'information sémantique en plus d'information syntaxique. Voici comment ils mettent leur information sémantique. Les prédicats sémantiques = dénoter les relations entre participants d'un évènement et l'évènement en soi. Ils sont utilisés pour transmettre les key components of meaning d'une classe verbale.

Dissertation : Les prédicats sémantiques sont utilisés pour véhiculer des composantes sémantiques importantes de chaque classe verbale. Ces prédicats sémantiques dénotent les relations entre les participants et les évènements. Dans verbnet, l'information sémantique est exprimée par une conjonction de prédicats sémantiques. Ces prédicats sémantiques peuvent ...

Revoir cette section, peut-être demandé à François de la lire, pour m'aider à comprendre.

LISTING 1.4. Les prédicats sémantiques

```
<SEMANTICS>
            <PRED value="cause">
                <ARGS>
                    <ARG type="ThemRole" value="Agent"/>
                    <ARG type="Event" value="E"/>
                </ARGS>
            </PRED>
            <PRED value="contact">
                <ARGS>
                    <ARG type="Event" value="during(E)"/>
                    <ARG type="ThemRole" value="?Instrument"/>
                    <ARG type="ThemRole" value="Patient"/>
                </ARGS>
            </PRED>
            <PRED value="degradation_material_integrity">
                <ARGS>
                    <ARG type="Event" value="result(E)"/>
                    <ARG type="ThemRole" value="Patient"/>
```

```
                    </ARGS>
                </PRED>
                <PRED value="physical_form">
                    <ARGS>
                        <ARG type="Event" value="result(E)"/>
                        <ARG type="VerbSpecific" value="Form"/>
                        <ARG type="ThemRole" value="Patient"/>
                    </ARGS>
                </PRED>
            </SEMANTICS>
        </FRAME>
```

1.1.2.6. *Exemples*

Chaque cadre syntaxique est accompagné d'un exemple pour exemplifier la construction syntaxique. On peut ainsi mieux cerner comment le verbe fonctionne.

Listing 1.5. Exemple

```
<EXAMPLE>Tony broke the window.</EXAMPLE>
```

1.1.3. Mapping de VerbNet à d'autres ressources lexicales

Les auteurs de VerbNet ont voulu mapper leur dictionnaire verbal à d'autres ressources lexicales de NLP. Ceci dans le but de permettre aux usagers d'utiliser les ressources linguistiques computationnelles disponibles et qu'il y ait des ponts entre celles-ci. À la base, ils ont fait ça car ils ont vu qu'il y avait beaucoup de gens entraînant des systèmes supervisés pour faire des analyses sémantiques, tel du word sense tagging et du semantic role labeling. Syntactic frames avec les tree frames dans Xtag, les verbes avec leur équivalent dans WordNet, les verbes et leur équivalent dans FrameNet II frame.

Le mapping à WordNet : Chaque verbe dans VerbNet est associé à un WordNet ID qui le lie, s'il existe une entrée pour ce lexème dans WordNet. Par exemple, en faisant le mapping de VerbNet à WordNet on voit que les sens représentés par le fait qu'un membre apparaisse dans différentes classes (faisant la désambiguisation) est aussi refletée dans l'occurrence du verbe dans de nombreux synsets représentant ces distinctions de sens. Les différences que VerbNet dans la sémantique changeante d'un verbe en fonction du contexte dans lequel on l'emploie se retrouve aussi dans l'un des synsets attribué à cette entrée lexicale. De plus, les classes verbales qui ont des caractéristiques sémantiques similaires vont pointer vers les mêmes SynSets. Ansi, puisque WordNet est un des lexicons les plus utilisés, le mapping à VerbNet le rend encore plus utile, car maintenant, on a accès à une interface sémantique-syntaxe et une liste de constructions syntaxiques et des prédicats sémantiques qu'on n'avait pas accès en aussi grande quantité auparavant.(Insérer l'exemple de VerbNet avec LEAVE)

Le mapping à FrameNet : à revoir, lire sur FrameNet pour mieux comprendre

Le mapping à XTAG : Ils ont fait un mapping entre les constructions syntaxiques et les arbres de XTAG. à revoir aussi, pas certain de comprendre le lien qui les unie. Revoir XTAG pour mieux comprendre

1.1.4. Utilisation de VerbNet dans des applications NLP

1.1.5. Brèves descriptions d'autres dictionnaires

Dorr's LCS database : La base de donnée de Dorr LCS,inclue une grande quantité de verbes organisés en classes sémantiques dérivés des classifications de Levin. Les auteurs de VerbNet trouvent qu'il y a un manque du côté des structures syntaxiques qui sont beaucoup trop simplifiées, et du fait que le lien entre la sémantique et la syntaxe n'est pas clair. Et une grande partie de ces données avaient été cherchées automatiquement menant à des incohérences, c'est pourquoi, après avoir fait une analyse des données ils n'ont pris que très peu d'information provenant de LCS.

The generative lexicon : Ce lexicon inclut une représentation syntaxique et sémantique des verbe ainsi qu'une représentation de la structure des évènements. VerbNet a une situation équivalente dans ses frames sémantiques qui incluent de l'information sur le déroulement de l'évènement (le prédicat) avec des fonctions temporelles.

Corelex : Ne traite que des noms

Comlex : ne fait pas de distinction entre les différents sens d'un verbe, et ne traite pas de sémantique explicitement. Mais il s'agit d'une base de données sur les constructions syntaxiques des unités lexicales qu'ils traitent.

Verbalex : fortement inspiré de VerbNet

GREG : utilise le formalisme DATR,

DDO : dictionnaire danois utilisant LFG pour faire son lexicon

LEFFF : un dictionnaire morpho-syntaxique pour le français

LEXSCHEM :

1.1.6. Pourquoi on n'a pas utilisé les rôles thématiques et les prédicats sémantiques

Parler de ça dans la section python où on crée les patrons de régime, ça s'y prêtre plus. p.210 dans le livre Melcuk on pourrait garder les rôles pis les mettre dans MATE aussi

1.1.7. Évaluation du système

chapitre 5, p.75 C'est important de parler de l'évaluation du système car c'est celui que nous utiliserons, et nous voulons voir les problèmes qu'ils ont rencontré.

D'ailleurs, les framesets de Propbank pourraient être très bénéfiques pour le dictionnaire, car il y a aussi les entrées nominales avec des fonctions lexicales pas explicites. Pour un travail futur.

1.2. Python

Pour cette section : faire des commentaires directement dans le code, et après, sous le code, expliquer ce que le code fait ensemble.

Tel que mentionné dans le section précédente concernant l'architecture de VerbNet. Nous avons vu comment les documents XML sont arrangés. Ainsi, à l'aide du module *xml.etree.cElementTree* nous avons pu faire des opérations sur l'ensemble des données de VN.

1.2.1. Création du dictionnaire de verbe : lexicon.dict

Listing 1.6. code pour lexicon.dict

```
def supers(t, i):
    ID = t.get('ID')
    sc = {ID:i}
    subclasses = t.findall('SUBCLASSES/VNSUBCLASS')
    if len(subclasses) > 0:
        for sub in subclasses:
            sc = {**sc, **supers(sub, ID)}
    return sc

def treeframes(t):
    ID = t.get('ID')
    z = []
    for frame in t.findall('FRAMES/FRAME'):
        description = re.sub(r"\s*[\s\.\-\ +\\\/\(\)]\s*", '_',
                frame.find('DESCRIPTION').get('primary'))
        if description in exclude:
            continue
        description = re.sub('PP', 'PP_{}', description)
        preps = [p.get('value') or p.find('SELRESTRS/SELRESTR').get('type').upper() for p in f
        preps = [sorted(p.split()) for p in preps]
        examples = [e.text for e in frame.findall('EXAMPLES/EXAMPLE')]
        if len(preps)==1:
            description = description.format('_'.join(preps[0]))
        elif len(preps)==2:
            description = description.format('_'.join(preps[0]), '_'.join(preps[1]))
```

```
        elif len(preps)==3:
            description = description.format('_'.join(preps[0]), '_'.join(preps[1]), '_'.join(
        z.append((description, examples))

    subclasses = t.findall('SUBCLASSES/VNSUBCLASS')
    subframes = [treeframes(subclass) for subclass in subclasses]
    subframes = sum(subframes, []) # flatten list of lists
    return [(ID, z)] + subframes

with open('lexicon.dict','w') as f:
    f.write('lexicon {\n')
    for file in [f for f in os.listdir('verbnet') if f[-4:] == '.xml']:
        root = ET.parse('verbnet/'+file).getroot()
        d = dict(treeframes(root))
        sc = supers(root, 'verb')
        for c in d.keys():
            f.write('"'+c+'"')
            if sc[c] == 'verb':
                f.write(': ' +sc[c] + ' {')
            else :
                f.write(': ' +'"'+sc[c]+'"' + ' {')
            [f.write('\n  gp = { id=' + gp[0] + (max(len(gp[0]), 30)-len(gp[0]))*' ' + ' dia=x
            f.write('\n}\n')
    f.write('\n}')
```

Dans les feuilles XML de VerbNet, l'information que nous cherchions pour améliorer notre système MATE était : tous les patrons de régimes possibles pour une classe de verbe. Au départ, nous ne savions pas encore ce que nous voulions extraire de ces feuilles XML, elles abondent en information. Toutefois, il semblait évident que nous pourrions vraiment en tirer parti. Nous avons donc commencé par extraire l'information se trouvant sous SYNTAX en pensant que le syntactic frame était ce que nous utiliserions pour construire notre lexicon. Toutefois, nous nous sommes vite rendu compte dans le processus que ce n'était pas exactement ce que nous voulions. Ça nous est apparu évident lorsque nous avons extrait les descriptions des syntactic frames, les exemples, puis les données sous SYNTAX. En effet, *NP V S INF ['NP', 'VERB', 'NP'] I loved to write.* un exemple comme celui-ci nous montre que les informations sous SYNTAX ne corresponde pas à ce que nous cherchions, car la description du patron de régime et le patron de régime en soi diffèrent. Ils mettent que "to write" correspond à un NP. Ce qui n'est pas le cas quand on y pense bien. De plus, lorsqu'on extrait les balises sous SYNTAX, on a décidé de ne pas extraire les attributs contenus sous les balises car c'était de l'information soit sur les rôles thématiques ou sur les prépositions.

Nous avons donc utilisé VerbNet de mieux que nous le pouvions. Ainsi, les descriptions des FRAMES SYNTACTIC étaient "accurates" et allait nous donner les brèves descriptions des patrons de régime que nous ajouterons à notre lexicon. Ainsi, pour l'exemple mentionné NP V S INF, ce que nous en retenions, c'est que pour cette classe de verbe, il existe un

patron de régime où on a un sujet, puis un verbe, et ensuite une proposition infinitive comme complément d'objet direct. Cette information était suffisante pour créer le dictionnaire de patron de régime. Car, nous pensions au départ que nous pourrions tout prendre de verbnet pour créer notre lexicon. Nous voulions chercher les descriptions des patrons de régime ainsi que l'information sous SYNTAX sous FRAME pour ainsi créer les patrons de régime en soi en les traduisant dans notre langage (TST). Toutefois, leur manière d'encore les patrons de régime ne correpond pas à la notre sur beaucoup trop d'aspect. Nous le verrons plus tard, il y a de la redondance à certains moments et notre théorie rend mieux compte des patrons de régime. Mais pour l'instant, ce qui est important de noter c'est que nous notons les patrons de régime en attribuant des actants I et des noms de relation pour signifier que est le rôle de cet actant lors du passage de la sémantique à la syntaxe. VerbNet ne fait pas cela de la même manière que nous. Ils ont aussi un volet sémantique, mais qui ne se branche pas à notre modèle théorique. Nous voulions donc nous inspirer quand même de leur méthode pour nos patrons de régime, mais le fait qu'ils utilisent les rôles thématiques nous posait problème. Il était difficile d'associer un rôle thématique à un actant syntaxique (bien que nous ayons tenté [montrer le graphique de F.L]). Donc, nous en sommes venus à la conclusion qu'il nous faudrait créer les patrons de régime en Python pour ensuite les exporter dans un format adéquat pour MATE. Pour ce faire, nous devions utilisé les descriptions des patrons de régime qui se retrouve dans *VerbNet* et à partir de la description, créer une fonction qui nous permettrait de générer rapidement des patrons de régime adéquat pour MATE en ayant simplement à remplir les trous.

Pour extraire les descriptions des patrons de régime nous avons utilisé deux fonctions. D'abord la fonction *treeframe* qui s'occupe de récupérer la description de chaque frame syntaxique à travers tout VN. Nous passons à travers tous les frames existant dans les XML de VN. Autant dans les classes que les sous-classes. Toutefois, cette fonction ne fait pas que récupérer la description du syntactic frame de VN et nous la recrache tel quel. Nous faisons quelques opérations sur les descriptions que nous extrayons. Notamment, nous remplaçons tout espace,tiret, point,barre oblique, paranthèses qui pourraient se trouver dans les descriptions par des _ à l'aide d'expressions régulières. Puis, nous retirons certaines descriptions de gp à cause de leur caractèrere problématique à encoder(pour des raisons théoriques, logiques – à expliquer ailleurs).Puis une fois qu'un clean up a été fait des descriptions et que nous avons uniquement celles que nous voulions, nous procédons à une seconde étape de raffinement des descriptions. Nous utiliserons une seconde fois une expression régulière pour trouver tous les occurrences de PP car nous ajouterons les prépositions impliquées dans les PP comme tel. Pour ce faire, nous faisons un search de tous les prépositions existant dans les patrons de régime de verbnet(et non dans la description, c'est là que les patrons de régime de

VN nous ont été utiles) et nous mettons les prépositions que nous soutirons des gp directement dans le nom de la description des gp à la suite du mot PP à chaque fois qu'on retrouve le mot PP dans une description. On obtient ainsi les descriptions nettoyées de celle qu'on ne veut pas, avec uniquement des underscore pour séparer les syntagmes puis les prépositions (lorsqu'il y a lieu) dans les noms des descriptions de gp.

Puis, une fois que nous faisions ces opérations sur les classes de VerbNet, nous avons scripté une méthode pour que la fonction s'applique aussi aux sous-classes.

Par la suite, nous avons créé une fonction *super* afin que la création du lexicon s'agence bien avec la manière que MATE fonctionne. Cette fonction nous permet d'utiliser le mécanisme d'héritage qui existe dans MATE. Ainsi, on peut renvoyer une sous-classe à la classe(ou sous-classe) qui la domine. De plus, cette fonction limite le nombre de descriptions de gp. VerbNet a aussi ce mécanisme d'intégrer autrement. Ainsi, si une classe X a 10 descriptions et qu'une sous-classe Y en a 5, mais que les 10 descriptions de la classe s'appliquent aussi à la sous-classe, on n'aura pas 15 descriptions mais juste 5 dans le sous-classe, car le mécanisme d'héritage s'occupe de faire ça. On a aussi programmé la création den notre lexicon pour que si il n'y a pas de sous-classes, que la classe hérite de la classe verb afin d'avoir les attributs de base de cette classe (la dpos, la spos, voir MATE). De cette manière, tous les classes héritent de la classe verb si on remonte aux classes mères, ainsi, on n'a pas besoin de préciser à chaque fois la dpos/spos.

Puis finalement, nous utilisons la fonction write qui écrit le tout dans un fichier .dict. Nous loopons à travers tous les fichiers compris dans le dossier VN. Puis nous allons chercher les keys et values du dictionnaire créé à l'aide de la fonction treeframe et du dictionnaire super.(À revoir)

Nous avons aussi pris la peine d'extraire les exemples pour chaque description car il s'agit d'une information utile pour voir dans quel contexte s'utilise ce patron de régime. Et c'est une information pertinente à avoir dans un dictionnaire, donc nous l'ajouterons à notre lexicon pour notre système de GAT. Pour ce faire, nos opérons de la même manière que pour les descriptions de gp dans le sens où nous faisons l'extractiond exemples en passant par les feuilles XML et le module d'extraction xml.etree. Pour chaque frame, nous passons la fonction à l'entièreté de VN et nous nous arrêtons à la balise EXAMPLES puis à tout texte se trouvant dans cette balise. Nous avons voulu extirper les exemples alongside des descriptions car c'est de l'information pertinente à avoir dans un dictionnaire.

1.2.2. Création du dictionnaire de patron de régimes

Listing 1.7. code pour gpcon.dict

```
def roman(n):
    return ['I', 'II', 'III', 'IV', 'V', 'VI'][n-1]

def gp(name, real_actant):
    s = name + ' {\n'
    i=0
    for actant in real_actant:
        i = i+1
        if type(actant) == list:
            for y in actant:
                s = s + "   " + roman(i) + "={" + y + "}\n"
        else:
            s = s + "   " + roman(i) + "={" + actant + "}\n"
    s = s + '}\n'
    return s


#SUBJECTIVE
subj = 'rel=subjective dpos=N'

#DIRECT OBJECTIVE
dir_N = 'rel=dir_objective dpos=N'
dir_V_ING = 'rel=dir_objective dpos=V finiteness=GER'
dir_V_INF = 'rel=dir_objective dpos=V finiteness=INF'

#INDIRECT OBJECTIVE
to_N = 'rel=indir_objective dpos=N prep=to'
indir_N = 'rel = indir_objective dpos = N'

#OBLIQUE
on_V = 'rel=oblique dpos=V prep=on'
to_obl_N = 'rel=oblique dpos=N prep=to'
for_obl_N = 'rel=oblique dpos=N prep=for'
as_N =  'rel=oblique dpos=N prep=as'
against_N = 'rel=oblique dpos=N prep=against'
at_N = 'rel=oblique dpos=N prep=at'

# LOC

locab = 'rel=oblique dpos=N prep=locab'
locad = 'rel=oblique dpos=N prep=locad'
locin = 'rel=oblique dpos=N prep=locin'

descriptions = {
'NP_agent_V' : [subj],
'NP_agent_V_NP' : [subj, dir_N],
'NP_asset_V_NP_PP_from_out_of' : [subj, dir_N, [from_N, out_of_N]],
'NP_attribute_V' : [subj],
'NP_attribute_V_NP_extent' : [subj, dir_N],
'NP_attribute_V_PP_by_extent' : [subj,by_N],
'NP_cause_V_NP' : [subj, dir_N ],
'NP_instrument_V_NP' : [subj, dir_N],
'NP_location_V' : [subj],
'NP_location_V_NP_theme' : [subj,dir_N],
```

```
'NP_location_V_PP_with_agent' : [subj, with_N],
'NP_location_V_PP_with_theme' : [subj, with_N],
'NP_material_V_NP' : [subj,dir_N],
'NP_material_V_PP_into_product' : [subj, into_N],

# CR\'{e}ATION DU GPCON
with open('gpcon.dict','w') as f:
    f.write('gpcon {\n')
    for d in descriptions.keys():
        f.write(gp(d, descriptions[d]))
    f.write('}')
```

Rajouter aussi en lstlistings les résultats qui sont des documents .dict (à la fin de l'explication)

Soit le mentionner ici, ou ailleurs, mais il a fallu faire un dictionnaire de patron de régime. D'abord, parce qu'on s'est rendu compte que du à toute l'information qu'on allait chercher et la différence dans le type d'information, on a jugé bon de créer un second dictionnaire qui ne contiendrait que les gps, autrement dit un gpcon. Celui l'information sur les patrons de régime (les actants syntaxiques). Il existe x nombre de gps répertoriés. Nous les avons trouvé en faisant un ensemble à partir de tous les descriptions que nous avons obtenus avec le script précédent. Une fois que nous avons l'ensemble des gps différents. Il nous fallait les créer, car tel que mentionné, nous ne pouvions pas extraire les gps de VerbNet dû à une différence trop grande (cadre théorique et application). Notre système de GAT fonctionne avec la théorie Sens-Texte et nous pensons que c'est la théorie qui s'y prête le plus pour faire ce type d'opérations et qui tient le mieux compte de la manière dont le langage fonctionne. Ainsi, nous avons créer le gpcon à partir de Python car un bon nombre d'opérations peuvent être automatisés (éviter les fautes, et c'est plus transparent). Pour la création du gpcon, notre dictionnaire en Python ressemblait à ça. Nos keys() étaient la description du gp et les valeurs étaient les actants syntaxiques impliqués dans ce gp (avec de l'information sur les actants syntaxiques nécessitant une préposition à réaliser). Selon l'ordre dans lequel figure nos objets dans la liste qui est ce qu'on retrouve dans values(), notre fonction va assigner le bon actant syntaxique(I, II, III, etc.) ainsi, cette partie est automatisée grâce à cette fonction. Après, pour l'objet "subj" on va lui assigner une string 'rel=subjective dpos=N' ce qui est encodé dans une autre cellule. Ainsi à chaque fois qu'un gp a un subj, on n'a pas à écrire ce que subj contient. Alors pour l'objet subj, on aura I et 'rel=subjective dpos=N'. C'est l'union de la fonction gp et de la fonction roman qui nous permettent d'assigner les bons actants syntaxiques aux objets dans la liste qui représente les values dans mon dictionnaire de gpcon.

1.2.3. Extraction des membres des classes verbales pour enrichir le dictionnaire

LISTING 1.8. code pour ajouter des lexèmes à lexicon.dict

```
def treemember(t):
    ID = t.get('ID')
    members = [m.get('name') for m in t.findall('MEMBERS/MEMBER')]
    subclasses = t.findall('SUBCLASSES/VNSUBCLASS')
    submembers = []
    if len(subclasses) > 0:
        for sub in subclasses:
            submembers = submembers + treemember(sub)
    return [(ID, members )] + submembers

files = [f for f in os.listdir('verbnet') if f[-4:] == '.xml']

members = dict(sum([treemember(ET.parse('verbnet/'+file).getroot()) for file in files], [])) #
values = sum(list(members.values()), []) # ici c'est uniquement les membres, sans infos sur le
dups = {m:[ID for ID in members.keys() if m in members[ID]] for m in values if values.count(m)
unique_member = {m:ID for ID in members.keys() for m in values if m in members[ID] and values.
lexemes = {d[0]+'_'+str(n+1):d[1][n] for d in dups.items() for n in range(len(d[1]))}

# Ici, j'ai fusionne les dictionnaires ensemble
unified_dict = {**unique_member, **lexemes}

with open('members.dict','w') as f:
    f.write('members {\n')
    for key in sorted(unified_dict.keys()):
        f.write(key)
        f.write(' : '),
        f.write('"'+str(unified_dict[key])+'"')
        f.write('\n')
    f.write('\n}\n')
```

1.2.4. Scripts pour faire les tests

1.2.4.1. *Extraction des exemples*

LISTING 1.9. code pour créer phrases.txt

```
def treeframes(t):
    z = []
    for frame in t.findall('FRAMES/FRAME'):
        description = re.sub(r"\s*[\s\.\-\ +\\\/\(\)]\s*", '_', frame.find('DESCRIPTION').get(
        if description in exclude:
            continue
        examples = [e.text for e in frame.findall('EXAMPLES/EXAMPLE')]
        z =  z + examples
    subclasses = t.findall('SUBCLASSES/VNSUBCLASS')
    subframes = [treeframes(subclass) for subclass in subclasses]
    subframes = sum(subframes, []) # flatten list of lists
    return z + subframes

liste=[]
with open('phrases.txt','w') as f:
    for file in [f for f in os.listdir('verbnet') if f[-4:] == '.xml']:
```

```
        root = ET.parse('verbnet/'+file).getroot()
        d = (treeframes(root))
        final_liste = liste + d
        [f.write(x+'\n') for x in final_liste]
```

1.2.4.2. *Création des structures qui serviront de tests*

Dans la figure ci-bas, on explique comment on a créer les documents .str qui serviront d'input à notre système MATE qui prend ce genre de document en entrée.

LISTING 1.10. code pour créer des structures .str

```
phrases = open('phrases.txt','r')

with open('structures.str','w') as f:
    for(i,p) in enumerate(phrases):
        with open('s'+str(i)+'.str','w') as g:
            structure = 'structure Sem S'+str(i)+'{\n S {text="'+p.strip()+'"\n\n main-> \n }\
            f.write(structure)
            g.write(structure)
```

Chapitre 2

GÉNÉRATION AUTOMATIQUE DE TEXTE

Pour cette partie, prendre l'article en 8 pages de François, et observer comment il décrit le déroulement de MATE. Reprendre l'idée, mais approfondir chaque point

Ce sera une section du mémoire sur le mécanisme dsynt=X

2.1. Mécanisme dsynt=created->constrained->OK pour générer un énoncé simple (sujet, verbe, objet)

2.1.1. Application de la règle root_standard

Cette règle crée un noeud qui sera la racine de l'arbre syntaxique. Ce noeud se fait imposer des contraintes. Notamment, on demande à ce que ce soit un lexème appartenant à la partie du discours : verbe et que sa finitude soit de type : fini. On impose à ce noeud le trait dsynt=constrained pour que ça s'harmonise avec le règle lex_standard, mais c'est à revoir.

2.1.2. application de la règle lex_standard pour lexicaliser le verbe principal

On assigne un lexème à un noeud créé en syntaxe profonde. Dans le contexte actuel, on l'utilise pour sélectionner l'unité lexicale qui matche les contraintes énoncées sur le noeud vide créé par la règle root_standard. Ce lexème provient du lexicon. Lorsque la règle s'applique et qu'elle consomme le noeud en y mettant la bonne lexicalisation (qui respecte les contraintes sur le noeud), on ajoute un trait dsynt=OK pour signifier que le sémantème a été réalisé en syntaxe profonde et qu'on ne fasse plus d'opérations sur ce noeud.

2.1.3. Application de la règle actant_gp_selection

Cette règle s'applique lorsque nous avons un prédicat. On crée une variable[?GP] qui nous fournit un chemin vers l'information encodée sous l'attribut *gp* d'un lexème [?X]. Puis , on extirpe les traits *id* et *dia* pour chaque attribut *gp* de notre [?X]. Une fois qu'on a récupéré ces informations, on les appose au lexème en question car on se servira de ces informations pour l'application de règles subséquentes. Le trait *id* représente la description du patron de régime (chaque description se retrouve dans notre gpcon qui est un dictionnaire de *gp*) et le trait dia nous renseigne sur la diathèse de ce patron de régime, c'est-à-dire combien d'actants sont en jeu, et dans quel ordre sont-ils placés ? Il est essentiel qu'un lexème verbal aille chercher ces traits car il en a besoin pour appliquer les règles actancielles qui en découlent. Il faut que le système sache quel patron de régime utilisé pour un prédicat donné, et dans quel ordre les actants seront réalisés en syntaxe.

2.1.4. application des règles actancielles

Une fois qu'un gp est sélectionné, on appliquera la règle actancielle qui lui correspond. Nos règles actancielles ressemblent à : actant_gp_ijk. Les règles prennent en input les arcs sémantiques liant les actants à leur prédicat. Ces règles génèrent des noeuds vides auxquels on appose un trait dsynt=created pour signifier qu'on vient de créer des noeuds vides (on dit qu'ils sont vides car ils n'ont pas encore été consommés par une unité lexicale) en syntaxe profonde. On obtient ainsi des arcs syntaxiques au bout desquels se trouve un noeud vide. Ces règles se font imposer des conditions bien strictes. Il faut d'abord que le prédicat qui les gouverne soit lexicalisé. Ce qui se traduisait par l'ajout d'un trait dsynt=OK au lexème lexicalisé (avec la règle lex_tandard). La règle actant_gp_selection nous permettait de soutirer les traits *id* et *dia* et c'est ici que le trait dia entre en jeu. On s'en sert pour illustrer la diathèse du verbe. [à revoir]

2.1.5. application de la règle constraints_gp

Cette règle applique des contraintes à des noeuds nouvellement créés (autrement dit, des noeuds qui ont le trait dsynt=created). C'est à partir de cette règle que le mécanisme de *dsynt=created-> constrained->OK* prend vie. Il s'agit d'une étape intermédiaire entre la création du noeud avec les règles actancielles et la lexicalisation qui consommera le noeud en octroyant un lexème suivant un trait dsynt=OK (signifiant que la réalisation en syntaxe profonde est terminée). Ainsi, la règle procède de cette manière. On prend un X (un prédicat) qui lie un Y (un actant) puis ce dernier se fait imposer des contraintes. On cherche à lui

imposer ces contraintes car on souhaite qu'il respecte le patron de régime de X. Ainsi, s'il ne convient pas comme actant syntaxique, il ne pourra pas passer à la lexicalisation. Car on souhaite une correspondance entre les traits naturels contenus dans le dictionnaire pour le sémantème en question. Si ses traits ne convienne pas aux contraintes imposées au noeud, provenant du patron de régime de X, alors l'arbre sera incomplet. Lorsqu'on met les contraintes sur le noeud, on lui met aussi le trait dsynt=constrained pour montrer qu'il a été contraint et donc qu'il remplit les critères pour passer à la lexicalisation.

2.1.6. application des règles de lexicalisation

Il y a différentes règles de lexicalisation afin de donner un peu de latitude au système. Ainsi, si quelques informations sont manquantes dans le lexicon ou le semanticon, nous voulons que le système arrive quand même à effectuer une génération de texte. C'est pourquoi il existe la règle standard qui fonctionne lorsque nous avons accès à tous les éléments nécessaires. Les régles de types *guess* opèrent lorsqu'il nous manque certaines infos.

2.1.6.1. *lex_standard*

Nous avons vu cette règle un peu plus haut, car il fallait l'appliquer dès le début pour lexicaliser le noyau principal afin de chercher le bon lexème qui pouvait remplir cette fonction. Une fois que nous l'avons lexicalisé, nous avons pu aller chercher les informations sur la nature de son gp etc. Nous sommes maintenant rendus au moment où il existe des arcs syntaxiques au bout desquels nous avons des noeuds contraints par des traits comme : la dpos, la finitude, la définitude, etc. Nous allons donc lexicaliser ces noeuds afin de poursuivre la construction de l'arbre, du haut vers le bas. En gros, comment ça fonctionne. On cherche la lexicalisation d'un sémantème donné dans le semanticon, en allant chercher le trait *lex* du sémantème. Puis, on colle cette lexicalisation a un noeud déjà existant. Ce noeud existe déjà car il a soit été créé par root_standard ou par une des règles actancielles. On va octroyer au noeud 3 traits : un trait dlex, un trait dpos, et finalement un trait dsynt. Le trait dlex est la lexicalisation profonde, celle qu'on retrouve dans le lexicon, le trait dpos est la partie du discours profonde qui doit correspondre à la dpos demandée par constraints_gp. Et finalement, un trait dsynt=OK qui s'ajoute à la lexicalisation du sémantème pour signifier au système que le noeud a été consommé et qu'il a été réalisé en syntaxe profonde. Donc, qu'on ne fasse plus d'opération sur ce noeud. Finalement, on peut uniquement faire des opérations sur des noeuds qui ont let trait dsynt=constrained afin de lexicaliser seulement les noeuds qui se sont fait attribués des contraintes. Ainsi, s'ils respectent les contraintes, ils pourront être lexicalisé, sinon, l'arbre sera incomplet et la génération échouera.

2.1.7. Avantages d'utiliser ces mécanismes

"Dsynt=OK" indique que le noeud a été consommé. Autrement dit, il existe maintenant une unité lexicale réalisé en syntaxe profonde, là où il y avait un noeud vide. "Dsynt=constrained" indique qu'un noeud vide s'est fait imposé des contraintes. Ces contraintes peuvent être de plusieurs ordres. Notamment, on impose une dpos, une finitude, un mood, etc. Il s'agit du passsage intermédiaire entre la création d'un noeud et sa lexicalisation. On veut s'assurer qu'il respecte certaines contraintes pour ne pas lexicaliser n'importe quoi lors de la génération de notre arbre syntaxique. Finalement, le trait dsynt=created

2.2. Énumérations

Voici une énumération avec numérotation :

1. item 1 ;
2. item 2 ;
3. item 3.

Maintenant, une énumération sans numérotation avec des marqueurs différents :

— Marqueur par défaut ;

• `$\bullet$` ;

⋆ `$\star$`.

2.3. Équations mathématiques

Une équation :

$$\otimes^n \mathbb{C}^2 \cong \bigoplus_{m=-n/2}^{n/2} W_m.$$

Une autre équation, cette fois-ci numérotée :

$$\frac{\partial \mathcal{L}}{\partial \phi^a} - \partial_\mu \frac{\partial \mathcal{L}}{\partial(\partial_\mu \phi^a)} = 0, \qquad \mu = 0,1,2,3. \tag{2.3.1}$$

Les équations (2.3.1) précédentes sont appelées *équations d'Euler-Lagrange* ou encore *équations du mouvement*. Dans les calculs suivants,

$$\begin{aligned} \delta S &= \int_\Omega \mathrm{d}^d x \, \mathcal{L}\Big(\phi'^a(x), \partial_\mu \phi'^a(x)\Big) - \int_\Omega \mathrm{d}^d x \, \mathcal{L}\Big(\phi^a(x), \partial_\mu \phi^a(x)\Big) \\ &= \int_\Omega \mathrm{d}^d x \, \left[\delta\phi^a \frac{\partial \mathcal{L}}{\partial \phi^a} + \partial_\mu \delta\phi^a \frac{\partial \mathcal{L}}{\partial(\partial_\mu \phi^a)}\right] \end{aligned}$$

$$= \int_\Omega \mathrm{d}^d x \left[(\delta\phi^a \frac{\partial \mathcal{L}}{\partial \phi^a} + \partial_\mu \left(\delta\phi^a \frac{\partial \mathcal{L}}{\partial(\partial_\mu \phi^a)} \right) - \delta\phi^a \partial_\mu \frac{\partial \mathcal{L}}{\partial(\partial_\mu \phi^a)} \right]$$
$$= 0,$$

aucune ligne n'est numérotée. Alors que dans ce qui suit, la dernière ligne l'est :

$$\begin{aligned} \delta S &= \int_{\Omega'} \mathrm{d}^d x' \, \mathcal{L}\Big(\phi'^a(x'), \partial'_\mu \phi'^a(x')\Big) - \int_\Omega \mathrm{d}^d x \, \mathcal{L}\Big(\phi^a(x), \partial_\mu \phi^a(x)\Big) \\ &= \int_\Omega \mathrm{d}^d x \left[\bar{\delta}\phi^a \frac{\partial \mathcal{L}}{\partial \phi^a} + \partial_\mu \bar{\delta}\phi^a \frac{\partial \mathcal{L}}{\partial(\partial_\mu \phi^a)} \right] + \int_{\partial\Omega} \mathrm{d}\sigma_\mu \, \mathcal{L}\Big(\phi^a, \partial_\mu \phi^a\Big) \delta x^\mu \\ &= \int_\Omega \mathrm{d}^d x \, \partial_\mu \mathcal{J}^\mu(x). \end{aligned} \tag{2.3.2}$$

2.4. Définitions, théorèmes et preuves

Voici une définition.

Définition 2.4.1 (La définition)**.** *La définition.*

Voici un théorème.

Théorème 2.4.1 (Titre)**.** *Ceci est vrai !*

Démonstration. Voici la preuve. □

Démonstration. Voici la preuve en gras. □

2.5. Construction d'un tableau

Tableau 2. I. Un tableau simple dans le second chapitre.

Option	g	c	d	`p{0.4\textwidth}`
Effet	À gauche	Au centre	À droite	Le texte de cette colonne est justifié et la largeur de la colonne est fixée à 40 % de la zone de texte (hors tableau).

Le tableau 2. I n'est pas très garni.

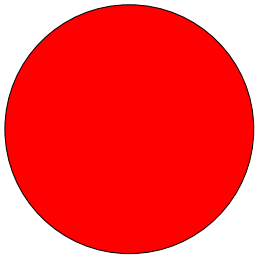
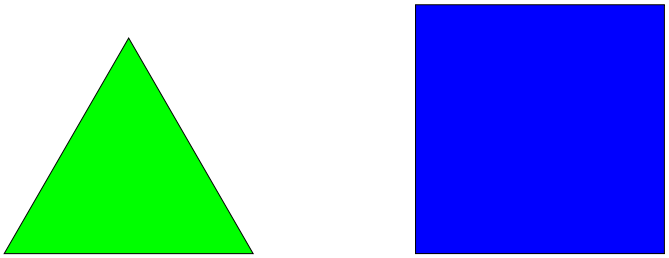


FIGURE 2.1. Un cercle.



(a) Un triangle. (b) Un carré.

FIGURE 2.2. Un carré et un triangle.

2.6. RÉFÉRENCE À UNE ENTRÉE BIBLIOGRAPHIQUE

Les documents par **??** ainsi que **?** sont des références en matière de LaTeX. Le manuel par **?** est probablement le plus populaire du lot.

L'article de **?** est, manifestement, très riche en rebondissements.

Les entrées du fichier `.bib` qui ne sont pas référencées dans le texte ne sont pas ajoutées à la bibliographie : un avantage de plus en faveur de BibTeX.

Dans ce paragraphe, on teste une cette référence **?**.

2.7. INSERTION DE FIGURES

La figure 2.1 est un *cercle*. À la figure 2.2, le triangle (a) et le carré (b) ont été placés côtes-à-côtes grâce à la commande `\subfigure`.

Bibliographie

Ayan, N. et B. J. Dorr. 2002, «Generating A Parsing Lexicon from an LCS-Based Lexicon», *LREC 2002 Workshop Proceedings : Linguistic Knowledge Acquisition and Representation : Bootstrapping Annotated Language Data*.

Baker, C. F., C. J. Fillmore et J. B. Lowe. 1998, «The Berkeley FrameNet Project», dans *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*, COLING '98, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 86–90, doi : 10.3115/980451.980860.

Grishman, R., C. Macleod et A. Meyers. 1994, «Comlex Syntax : Building a Computational Lexicon», dans *Proceedings of the 15th Conference on Computational Linguistics - Volume 1*, COLING '94, Association for Computational Linguistics, Kyoto, Japan, p. 268–272, doi :10.3115/991886.991931.

Kipper, K., H. T. Dang et M. Palmer. 2000, «Class-Based Construction of a Verb Lexicon», dans *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, AAAI Press, ISBN 978-0-262-51112-4, p. 691–696.

Kipper, K., A. Korhonen, N. Ryant et M. Palmer. 2006, «Extending VerbNet with Novel Verb Classes», dans *Proceedings of the Fifth International Conference on Language Resources and Evaluation – LREC'06 (http ://verbs.colorado.edu/ Mpalmer/Projects/Verbnet.Html)*.

Levin, B. 1993, «English verb classes and alternations : A preliminary investigation», .

Miller, G. A. 1995, «WordNet : A Lexical Database for English», *Commun. ACM*, vol. 38, n° 11, doi :10.1145/219717.219748, p. 39–41, ISSN 0001-0782.

Moens, M. et M. Steedman. 1988, «Temporal Ontology and Temporal Reference», *Comput. Linguist.*, vol. 14, n° 2, p. 15–28, ISSN 0891-2017.

Pustejovsky, J. 1991, «The Generative Lexicon», *Comput. Linguist.*, vol. 17, n° 4, p. 409–441, ISSN 0891-2017.

Schuler, K. K. 2005, «Verbnet : A Broad-coverage, Comprehensive Verb Lexicon», PhD Thesis, University of Pennsylvania, Philadelphia, PA, USA.

Annexe A

LE TITRE

A.1. Section un de l'annexe A

La première annexe du document.

Pour plus de renseignements, consultez le site web de la FESP. Pour plus de renseignements, consultez le site web de la FESP.

Tableau A. I. Liste des parties

Les couvertures conformes	obligatoires
Les pages de garde	obligatoires
La page de titre	obligatoire
Le résumé en français et les mots clés français	obligatoires
Le résumé en anglais et les mots clés anglais	obligatoires
Le résumé de vulgarisation	facultatif
La table des matières, la liste des tableaux, la liste des figures	obligatoires
La liste des sigles, la liste des abréviations	obligatoires
La dédicace	facultative
Les remerciements	facultatifs
L'avant-propos	facultatif
Le corps de l'ouvrage	obligatoire
L'index analytique	facultatif
Les sources documentaires	obligatoires
Les appendices (annexes)	facultatifs
Le curriculum vitæ	facultatif
Les documents spéciaux	facultatifs

Annexe B

LE TITRE2

texte annexe B