

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

12 mars 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

Sommaire	iii
Summary	v
Liste des tableaux	xi
Liste des figures	xiii
Liste des sigles et des abréviations	xv
Dédicaces	xvii
Remerciements	xix
Introduction	1
Chapitre 1. Génération automatique de texte	3
1.1. Contexte	3
1.2. Processus	4
1.3. Réalisation linguistique	4
1.3.1. Réalisateur surface	4
1.3.2. Réalisateur profond	4
1.3.2.1. GenDr :héritier de MARQUIS	4
1.4. Lexicalisation	4
1.5. Les verbes	4

1.5.1.	problématique	4
1.5.2.	les patrons de régime	4
Chapitre 2.	Un dictionnaire de patrons de régime : VerbNet	5
2.1.	VerbNet	6
2.1.1.	Classes verbales de Levin	6
2.1.2.	Composantes de VerbNet	8
2.1.2.1.	Classes verbales : organisation hiérarchique	9
2.1.2.2.	Membres	10
2.1.2.3.	Rôles thématiques	11
2.1.2.4.	Restrictions sélectionnelles	12
2.1.2.5.	Cadres syntaxiques	12
2.1.2.6.	Prédicats sémantiques	14
2.1.3.	Brèves descriptions de dictionnaires concurrents utilisés en NLP	15
2.1.3.1.	WordNet	15
2.1.3.2.	FrameNet	16
2.1.3.3.	XTAG	17
2.1.3.4.	Lexical conceptual structures	19
2.1.3.5.	Comlex	20
2.1.3.6.	LexSchem	20
2.1.3.7.	A large SCF lexicon for NLP apps : Valex	21
2.1.3.8.	VDE-Valency dictionary of English	22
2.1.3.9.	Dicovalence	23
2.1.4.	Utilisation de VerbNet dans des applications NLP	23
2.1.5.	Pourquoi on n'a pas utilisé les rôles thématiques et les prédicats sémantiques	24
2.1.6.	Évaluation du système	24

2.1.7. Pourquoi avoir choisi VerbNet ?	24
2.2. Python	25
2.2.1. Création du dictionnaire de verbes : lexicon.dict	25
2.2.1.1. Création du dictionnaire initial contenant les 278 classes verbales comme entrées lexicales	26
2.2.1.2. Extraction des 6393 membres des classes verbales pour enrichir le dictionnaire lexicon.dict	28
2.2.2. Création du gpcon	29
2.2.3. Scripts pour faire les tests	31
2.2.3.1. Extraction des exemples	31
2.2.3.2. Création des structures qui serviront de tests	31
Chapitre 3. Intégration des régimes de VerbNet dans GenDR	33
3.1. Comment générer du texte avec GenDR	33
3.1.1. Dictionnaires	33
3.1.2. Grammaires	33
3.1.3. Graphes	33
3.2. Implémentation des patrons de régime	33
3.2.0.1. GP dans les dictionnaires	33
3.2.0.2. GP dans les règles grammaticales	33
3.3. Mécanisme dsynt=created->constrained->OK pour générer un énoncé simple (sujet, verbe, objet)	33
3.3.1. Application de la règle root_standard	33
3.3.2. application de la règle lex_standard pour lexicaliser le verbe principal	34
3.3.3. Application de la règle actant_gp_selection	34
3.3.4. application des règles actanciels	34

3.3.5.	application de la règle constraints_gp.....	35
3.3.6.	application des règles de lexicalisation	35
3.3.7.	lex_standard.....	35
3.3.8.	Avantages d'utiliser ces mécanismes.....	36
Chapitre 4.	Évaluation du système.....	37
4.1.	Méthodes d'évaluation en NLG.....	37
Bibliographie	-i
Annexe A.	Le titre	A-i
A.1.	Section un de l'annexe A.....	A-i
Annexe B.	Le titre2	B-i

LISTE DES TABLEAUX

A. I	Titre alternatif pour la table des matières.....	A-i
------	--	-----

LISTE DES FIGURES

2.1	prédicats sémantiques	15
4.1	lexstd	37

LISTE DES SIGLES ET DES ABRÉVIATIONS

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

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 un bon fonctionnement des applications NLP.

Chapitre 1

GÉNÉRATION AUTOMATIQUE DE TEXTE

Branche du TAL but est de produire du texte automatiquement à partir de contenu pas en langue naturelle comme des données brutes, représentations, informations entreposées dans des bases de données (voir les inputs des différents systèmes)

Dans notre quotidien on voit des exemples de NLG tout simple, mais que faire quand on veut pousser la chose (voir la thèse intéressante). Les faciles sont des template-based text generation, on s'en fout. Ensuite y'a les rule-based generation et les trainable NLG

la gat créée pour répondre à des besoins traduction automatique (système de Dorr) résumés automatiques QUestion answering dialogue systems utilisé dans des domaines variés

mais variété entraîne un manque d'uniformité dans l'architecture, tâche, type de données (cité Lambrey). Reiter et Dale se sont penchés sur la question. Ont tenté de donner une ligne directive

Danlos et Reiter/Dale sur le processus de génération automatique de texte

Ils ont découpé les étapes importantes de la GAT de cette manière :

blabla bla

là-dedans, dans une étape importante, la réalisation linguistique, y'a 2 sous-étapes : lexicalisation et arborisation

ensuite la problématique : les verbes, les dictionnaires verbaux, pourquoi les verbes, l'avantage

couvrir large en GAT grâce aux verbes

1.1. CONTEXTE

utilité, manque d'uniformité, avenir,etc.

1.2. PROCESSUS

voir les thèses ou ouvrages qui parlent de NLG en général

Dale et Reiter pensent qu'il y a 6 tâches

Content determination Discourse planning sentence aggregation lexicalization referring
expression generation linguistic realization

1.3. RÉALISATION LINGUISTIQUE

1.3.1. Réalisateur surface

1.3.2. Réalisateur profond

FORGe : très similaire à ce qu'on fait, est-ce que c'est problématique ? En quoi on se démarque d'eux ?

1.3.2.1. *GenDr* : héritier de *MARQUIS*

article de François article sur MARQUIS

1.4. LEXICALISATION

1.5. LES VERBES

gross : constructing lexicon-grammar melcuk : ECD la richesse des verbes et pas facile à faire en NLG VerbNet a tenté de pallier à ça aussi

1.5.1. problématique

On n'est pas les seuls à penser que pour faire un bon rule-based grammar, on a besoin de ressources lexicales riches, dont des dictionnaires de sous-catégorisation pour être capable de générer toutes les phrases possibles en anglais (FORGE et l'article de towards large-coverage detailed lexical resources)

1.5.2. les patrons de régime

finir avec : il nous fallait trouver un dictionnaire capable de pallier à notre objectif, puis chapitre suivant

Chapitre 2

UN DICTIONNAIRE DE PATRONS DE RÉGIME : VERBNET

(ce début de chapitre sera probablement supprimé après la réorganisation du mémoire)

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 Théorie Sens-Texte (TST) qui servira à générer du texte. Cet objectif provient d'une problématique que nous avons 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. 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 les entrées de leur dictionnaire pouvaient s'appliquer en génération automatique de texte (GAT). De nos jours, avec les modèles stochastiques où il n'y a pas d'analyse linguistique, on ne construit plus 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 percevant les langues naturelles uniquement comme des suites de caractères. C'est une mode qui fonctionne présentement grâce à la quantité immense d'information en langue naturelle qui existe sur le web. En combinant ces nouveaux corpus incroyables avec

la puissance des ordinateurs et le développement en apprentissage machine, certains chercheurs font complètement fi de l'analyse linguistique dans leurs applications TAL et arrive à des résultats relativement bon (manque une citation ici). Toutefois, en ce qui concerne la GAT, le traitement de la langue se doit d'être impeccable (Lareau, Lambrey, Dubinskaite, Galarreta-Piquette et Nejat, 2018). D'où la nécessité de développer des outils puissant et rigoureux. Toutefois, les systèmes de GAT à base de règles sont plus coûteux car il faudra développer les règles de grammaire et se doter d'un dictionnaire assez large pour couvrir une ou des langues. Cependant, tout en étant bien conscient du changement de cap dans le domaine vers le *machine learning*, nous pensons qu'il est encore primordial de développer de bons outils linguistiques computationnels à base de règles et de dictionnaires car c'est de cette manière qu'on pourra le mieux représenter les langues naturelles.

Mentionner le blog de E.Reiter concernant les approches ML et la NLG (rule-based est meilleur, mais on pourrait se servir de ML, sans toutefois rely dessus à 100% comme certains chercheurs le font. source (

2.1. 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 saurait tenir compte de la richesse et la complexité que renferment les verbes (Kipper, Dang et Palmer, 2000). Les auteurs du projet trouvaient qu'il y avait un manque de lignes directrices sur l'organisation des verbes dans les dictionnaires destinés à des applications TAL. Malgré la quantité impressionnante de dictionnaires computationnels existant déjà, ils ont tout de même voulu créer un dictionnaire de verbe qui pourrait pallier à ce manque. (Faire un court retour sur les lacunes des autres dictionnaires)

2.1.1. Classes verbales de Levin

Le travail de Levin(Levin, 1993) a été de créer un dictionnaire où les verbes de la langue anglaise sont répartis dans une nombre fini de classes verbales. L'appartenance à une classe verbale est motivée par des comportements syntaxiques communs entre les verbes de cette classe. Son travail est le fruit d'observations sur les alternances de diathèses que démontrent les verbes. Levin remarquait que tout locuteur natif d'une langue est conscient des alternances de diathèses possibles d'un verbe, et ce sans avoir de connaissances linguistiques au préalable. Pour cette raison, elle a suivi son intuition et a tenté de délimiter tous les patrons de régime qu'un verbe possède, puis a évalué si d'autres verbes partageaient ces mêmes patrons. Lorsque c'était le cas, elle établissait une classe verbale qui allait regrouper les verbes

se comportant de la même manière. Bien que son travail s'insère dans le cadre de la syntaxe, elle postulait que les verbes qui se comportent de la même manière syntaxiquement possèdent aussi des propriétés sémantiques communes. Ainsi, comme ces verbes partagent les mêmes composantes sémantiques, il est normal que cela se reflète en surface via des comportements syntaxiques communs. Les composantes sémantiques sous-jacentes seraient à l'origine des comportements syntaxiques permis pour un verbe. Donc, des verbes qui partagent des comportements syntaxiques, partagent aussi des composantes sémantiques, mais ça ne veut pas dire que les verbes appartenant à la même phrase signifient la même chose. Cela veut dire qu'ils possèdent des caractéristiques sémantiques similaires. Levin met en garde que deux verbes synonymiques peuvent très bien appartenir à deux classes différentes tout comme deux verbes qui en apparence ne se ressemblent pas du tout, peuvent très bien partager des composantes sémantiques similaires.

Un avantage de regrouper les verbes en classes verbales, à part la valeur théorique qui était de démontrer les propriétés sémantiques communes à ces verbes dégage aussi un côté pratique qui était de construire un dictionnaire où les entrées lexicales ne sont pas prises individuellement, mais regroupée pour faciliter l'ajout d'entrées lexicales. Lorsqu'on termine le traitement d'une entrée, on n'a pas besoin de décrire tous les patrons de régime associés, on n'a qu'à ajouter l'entrée dans la classe qui la représente. Les auteurs de VerbNet notent toutefois que le classement de certains verbes est un peu tiré par les cheveux, mais ils ont revisité le classement initial de Levin et y ont apporté quelques modifications(Schuler, 2005).

Voici un exemple tiré de la thèse de Schuler (Schuler, 2005) qui nous démontre l'idée derrière la construction du dictionnaire de Levin. On prend les verbes *break* et *cut*, puis on teste diverses configurations possibles pour confirmer s'ils appartiennent à la même classe ou bien à deux classes distinctes. On pourrait penser qu'ils appartiennent à la même classe compte tenu de leur signifié qui se ressemblent. Briser et découper partagent évidemment des composantes sémantiques car il y a le sens d'altérer quelque chose, mais le court exemple nous démontre qu'ils appartiendrait à deux classes distinctes.

(1) *Transitive construction*

- a. John broke the window.
- b. John cut the bread.

(2) *Middle construction*

- a. Glass breaks easily.
- b. This loaf cuts easily.

(3) *Intransitive construction*

- a. The window broke.
- b. *The bread cut.

(4) *Conative construction*

- a. *John broke at the window.
- b. John valiantly cut at the frozen loaf, but his knife was too dull to make a dent in it.

On voit d'abord que les constructions en (1) et en (2) sont possibles pour ces deux verbes. Toutefois, en (3) et en (4), on remarque qu'ils ne partagent pas ces cadres syntaxiques. *Break* prend seulement la construction intransitive et exclut la conative, tandis que *cut* prend la construction conative et exclut l'intransitive. Selon la logique de Levin, cela est due à des différences de composantes sémantiques. Le verbe *cut* décrit une série d'actions entreprises dans le but de séparer un objet en morceaux. Toutefois, il est possible de commencer à découper un objet sans que l'objet ne soit séparé. Dans ce scénario, on peut tout de même percevoir que l'objet a été découpé. En ce qui concerne *break*, le changement d'état, le fait d'être séparé en morceau est le cœur même de l'évènement. Si on n'arrive pas au résultat final, une tentative de briser quelque chose ne peut être perçue.

Le projet de Levin a inspiré beaucoup de chercheurs, notamment l'équipe de VerbNet. C'est pourquoi ils ont repris une grande partie du travail de Levin. On nommera l'organisation hiérarchique de VerbNet en classe et en sous-classes et le regroupement des verbes en classes verbales. Toutefois, les auteurs de VerbNet ont retravaillé les entrées de Levin et y ont apporté des corrections et améliorations pour que le traitement des verbes soit meilleur (Kipper, Korhonen, Ryant et Palmer, 2006).

2.1.2. Composantes de VerbNet

Comme le système de Levin, VerbNet est aussi organisé en classes verbales. Chaque classe contient un ensemble de membres, une liste de rôles thématiques (accompagnés de restrictions sélectionnelles) utilisés pour décrire les arguments, puis un ensemble de cadres syntaxico-sémantiques possibles pour une classe. Chaque cadre est composé d'une brève description, suivi d'un exemple, puis d'une description syntaxique et des prédicats décrivant le cadre en question (Schuler, 2005).

2.1.2.1. *Classes verbales : organisation hiérarchique*

Les auteurs de VerbNet se sont fortement inspirés de Acquilex Lexical Knowledge Base (Copestake, 1992) qui avait organisé leur information lexicale en hiérarchie. Effectivement, les auteurs de VerbNet ont implémenté l’aspect hiérarchique en créant jusqu’à trois niveaux de profondeur dans les classes verbales de Levin (Schuler, 2005). Ainsi, une sous-classe verbale hérite de tout le contenu lexical de la classe (ou de la sous-classe) qui la domine. Les sous-classes ont été créées pour spécifier qu’un sous-ensemble de verbes issus de la classe qui les domine démontrent des comportements différents du reste de la classe tout en étant des verbes qui partagent les restrictions de la classe dominée. (guidelines, (Schuler, 2005)). Les comportements différents comprennent : les constructions syntaxiques, les prédicats sémantiques et les restrictions sélectionnelles sur les rôles thématiques.

Prenons un exemple tiré de VerbNet pour en expliciter la hiérarchie.

- Spray-9.7
 - Spray-9.7-1
 - * Spray-9.7-1-1
 - Spray-9.7-2

Spray-9.7 est le nom de la classe et celle qui dominera toutes les autres sous-classes. À l’intérieur de celle-ci, on spécifie tous les membres appartenant à cette classe, les rôles thématiques, les cadres syntaxiques et les prédicats sémantiques. Puis *Spray-9.7-1* est une sous-classe fille qui hérite de l’information de sa mère, mais précise d’autres informations. Comme un sous-ensemble de verbes propres à ces comportements différents. Puis *Spray-9.7-1-1* est une sous-classe d’une sous-classe et la hiérarchie continue. Elle héritera des traits de sa classe mère ainsi que de la classe qui domine sa classe mère. Finalement *Spray-9.7-2* est la classe sœur de *Spray-9.7-1* donc, elle hérite aussi des traits de *Spray-9.7* mais ne partage pas les particularités de *Spray-9.7-1*

Tel que démontré dans l’exemple, les classes et sous-classes sont numérotées. D’abord, pour expliciter le lien hiérarchique qui transcende à l’intérieur d’une classe. Mais la numérotation est aussi directement héritée du système de Levin (Levin, 1993). De cette manière, les classes sont numérotées par des chiffres allant de 9-109 (guidelines). Le numéro associé à des classes sert à représenter le partage de caractéristiques sémantiques et syntaxiques entre les classes verbales. Par exemple, les classes signifiant ‘mettre quelque chose’ commenceront par le chiffre 9.

- 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

2.1.2.2. *Membres*

Ainsi, tel que mentionné précédemment, les entrées lexicales dans VerbNet sont des classes verbales. Contrairement à des dictionnaires où chaque entrée individuelle représente un verbe, ici on a une entrée lexicale qui représente une panoplie de verbes. Ce qui permet à VerbNet de couvrir très largement la langue anglaise dans un format adapté aux applications TAL. Ainsi, pour garnir leur section *Members*, qui regroupe les verbes appartenant à une classe verbale en question, les chercheurs ont puisé dans d'autres ressources lexicales dont la base de données LCS (Ayan et Dorr, 2002) pour enrichir leur lexique.

La figure suivante montre à quoi ressemble la section *Members* dans VerbNet. Chaque classe verbale représente un document XML qui contient toutes les sections sous formes de balises.

LISTING 2.1. les membres

```
<VNCLASS ID="give-13.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="vn_schema-3.xsd">
  <MEMBERS>
    <MEMBER name="deal "
      wn="deal%2:40:01 deal%2:40:02 deal%2:40:07 deal%2:40:06 "
      grouping="deal.04"/>
    <MEMBER name="lend "
      wn="lend%2:40:00 "
      grouping="lend.02"/>
    <MEMBER name="loan "
      wn="loan%2:40:00 "
      grouping=""/>
    <MEMBER name="pass "
      wn="pass%2:40:00 pass%2:40:01 pass%2:40:13 pass%2:38:04 "
```



```

        grouping="pass.04"/>
    <MEMBER name="peddle"
        wn="peddle%2:40:00"
        grouping="peddle.01"/>
    <MEMBER name="refund"
        wn="refund%2:40:00"
        grouping="refund.01"/>
    <MEMBER name="render"
        wn="render%2:40:02 render%2:40:01 render%2:40:00 render%2:40:03"
        grouping="render.02"/>
    <!--removed "trade" from class because doesn't take "to-PP"-->
    <!--removed "volunteer" from class because doesn't fit dative or-->
    <!--PP recipient PP frames-->
</MEMBERS>

```

Ainsi, dans cet exemple, on voit que *deal*, *lend*, *loan*, *pass*, *peddle* et *refund* sont les membres de la classe *give-13.1*.

2.1.2.3. Rôles thématiques

VerbNet emploie 23 rôles thématiques pour identifier les arguments sélectionnés par les verbes. Ils ont opté pour cette approche d'identification car elle permet d'ajouter de l'information sémantique sur les participants. Contrairement à une approche où on énumère les arguments *Arg-1 Verbe Arg-2* comme dans PropBank (Palmer, Gildea et Kingsbury, 2005). À la base, les rôles thématiques ont été mis de l'avant par Fillmore (Fillmore, 1968) et Jackendoff (Jackendoff, 1972) pour identifier les arguments en leur assignant un rôle sémantique. Chaque argument se fait donné un rôle unique.

VerbNet critiquait les autres dictionnaires verbaux qui n'offraient pas de contenu sémantique. C'est pourquoi ils font la promotion de leur aspect sémantique via la section *Semantic frames* et les *Thematic Roles* (Schuler, 2005). Concrètement, parmi les 23 rôles thématiques choisis par VerbNet, certains proviennent de Fillmore et Jackendoff, d'autres s'en inspirent. C'est pourquoi ils se justifient en disant que le chiffre 23, et la nature des rôles est effectivement arbitraire, mais utile à la construction de leur dictionnaire. Voici la liste des rôles thématiques qu'ils ont choisi : *actor*, *agent*, *asset*, *attribute*, *beneficiary*, *cause*, *location*, *destination*, *source*, *experiencer*, *extent*, *goal*, *instrument*, *material*, *product*, *patient*, *predicate*, *recipient*, *stimulus*, *theme*, *time*, *topic*. Ces rôles ne sont pas spécifiques à des classes en particulier, les auteurs voulaient des rôles pouvant identifier tous les arguments possibles dans leur corpus. Donc, des rôles assez génériques pouvant se prêter à divers cadres.

À l'intérieur de chaque classe verbale (et sous-classe si c'est nécessaire), les rôles thématiques en jeu y sont listés dans la section <THEMROLES>. Ils sont ensuite mappés aux

arguments sélectionnés dans les cadres syntaxiques et sémantiques (qu'on voit à la figure "cadres syntaxique").

LISTING 2.2. les rôles thématiques

```
<THEMROLES>
  <THEMROLE type="Agent">
    <SELRESTRS logic="or">
      <SELRESTR Value="+" type="animate"/>
      <SELRESTR Value="+" type="organization"/>
    </SELRESTRS>
  </THEMROLE>
  <THEMROLE type="Theme">
    <SELRESTRS/>
  </THEMROLE>
  <THEMROLE type="Recipient">
    <SELRESTRS logic="or">
      <SELRESTR Value="+" type="animate"/>
      <SELRESTR Value="+" type="organization"/>
    </SELRESTRS>
  </THEMROLE>
</THEMROLES>
```

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

2.1.2.4. Restrictions sélectionnelles

Les restriction sélectionnelles s'ajoutent aux rôles thématiques. Il s'agit de restrictions imposées aux rôles thématiques afin que certains types d'arguments soient sélectionnés. Ces traits fournissent encore plus d'informations sémantiques sur l'argument. Dans l'exemple fournit ici, on remarquera que l'Agent est de type animé ou une organisation.

LISTING 2.3. les restrictions sélectionnelles

```
<THEMROLES>
  <THEMROLE type="Agent">
    <SELRESTRS logic="or">
      <SELRESTR Value="+" type="animate"/>
      <SELRESTR Value="+" type="organization"/>
    </SELRESTRS>
  </THEMROLE>
```

2.1.2.5. Cadres syntaxiques

Pour une classe donnée, on y retrouve des cadres syntaxiques qui représentent le type de réalisation de surface possible. D'ailleurs, ceux-ci 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 caractéristique 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 réalisations 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 2.4. cadres syntaxiques

```
<SYNTAX>
  <NP value="Agent">
    <SYNRESTRS/>
  </NP>
  <VERB/>
  <NP value="Theme">
    <SYNRESTRS/>
  </NP>
```

```

        <PREP value="to">
            <SELRESTRS/>
        </PREP>
        <NP value="Recipient">
            <SYNRESTRS/>
        </NP>
    </SYNTAX>

```

2.1.2.6. *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

...

LISTING 2.5. Les prédicats sémantiques

```

<SEMANTICS>
    <PRED value="has_possession">
        <ARGS>
            <ARG type="Event" value="start(E)"/>
            <ARG type="ThemRole" value="Agent"/>
            <ARG type="ThemRole" value="Theme"/>
        </ARGS>
    </PRED>
    <PRED value="has_possession">
        <ARGS>
            <ARG type="Event" value="end(E)"/>
            <ARG type="ThemRole" value="Recipient"/>
            <ARG type="ThemRole" value="Theme"/>
        </ARGS>
    </PRED>
    <PRED value="transfer">
        <ARGS>
            <ARG type="Event" value="during(E)"/>
            <ARG type="ThemRole" value="Theme"/>
        </ARGS>
    </PRED>
    <PRED value="cause">
        <ARGS>
            <ARG type="ThemRole" value="Agent"/>
            <ARG type="Event" value="E"/>
        </ARGS>
    </PRED>

```

Pour mieux exposer leur sémantique, nous avons fait un tableau qui exemplifie les prédicats sémantiques pour l'énoncé : give

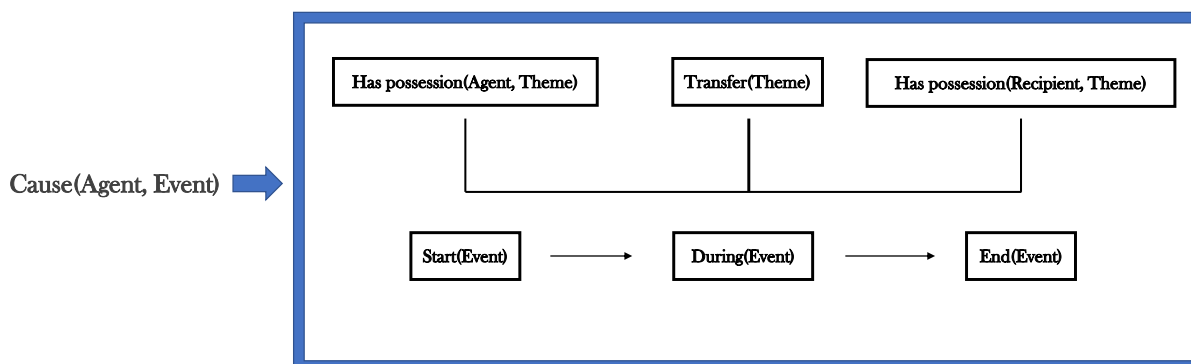


FIGURE 2.1. prédicats sémantiques

2.1.3. Brèves descriptions de dictionnaires concurrents utilisés en NLP

2.1.3.1. *WordNet*

: Wordnet est une base de données lexicale en ligne pour les verbes, noms, adjectifs et ad-
verbes de la langue anglaise. Cette base de donnée est organisée en ensemble de synonymes,
où chaque *synset* contient des mots référant à un concept donné ainsi qu'une glose défini-
tion et une phrase exemple. L'aspect hiérarchique est construit via des liens d'hyperonymie
et d'hyponymie entre les *synsets*. Dans WordNet, les verbes sont classés parmi les classes
suivantes s'il s'agit de verbes dénotant des actions ou des évènements (motion, perception,
contact, communication, competition, change, cognition, consumption, creation, emotion,
perception, possession, bodily care and functions, social behavior and interactions) et des
verbes dénotant des états (des types être :resemble, belong, suffice, des verbes contrôles
(want, fail, prevent, succeed), et des verbes aspectuels (begin)) (Fellbaum, 1998)

Les verbes sont groupés par ensemble de synonymes et non par classes verbales contrai-
rement à VerbNet. Le concept de synonyme n'est pas vu dans son sens typique, il s'agit
plus de : items lexicaux qu'on pourrait utiliser dans un même contexte sont vus comme des
synonymes. À la base WordNet a été conçu pour agir comme réseau sémantique et contient

ainsi peu d'information syntaxique. La ressource fournit des définitions, des exemples et des synsets, mais ne nous donne pas d'information sur la structure sémantique ou syntaxique des verbes, elle est sous-entendue, ce que fait que contrairement à VerbNet qui explicite en détail chaque patron de régime et structure syntaxique. les liens entre chaque items lexicaux sont de divers ordres : synonymes, antonymes, troponymes, entailment et causation.(Schuler, 2005)

"English verbs as a semantic net" (Fellbaum, 1998) : Dans cet article, on expose le liens sémantiques entre les verbes, ce que wordnet fait : il tisse un lien entre 2 verbes qui partagent des composantes sémantiques : ces liens sont de divers ordres et justifié par le concept d'entailment. Ainsi, on a les relations de hyponymie, hyperonymie, méronymie, troponymie, etc.

D'ailleurs, VerbNet a décidé de faire un mapping de ses entrées lexicales aux entrées lexicales de WordNet pour compléter la ressource. Chaque verbe dans VerbNet est mappé à WordNet si c'est possible et qu'il y existe un équivalent. On peut aussi confirmer le choix d'associer un mot à plusieurs classes verbales dans VerbNet. Ainsi, le verbe leave est associé à Escape, future having, keep, leave et fulfill car leave peut impliquer différentes composantes sémantiques en fonction des prédicats décrits dans ces classes. On retrouve l'équivalent dans les synset qui sont attribués à Leave (toutefois, WordNet attribue 14 différents sens à Leave) et VerbNet en attribue 5. Ainsi, lorsque c'est possible, les membres des classes se font assigner un WordNet ID avec.

2.1.3.2. *FrameNet*

: Le projet de Berkeley FrameNet contient de l'information sur les noms, adjectifs et verbes le tout basé sur un corpus annoté manuellement. Les sens des mots sont groupés dans des structures conceptuelles appelées des Frames qui partagent des propriétés sémantiques similaires et qui sont des représentations de concepts cognitifs. La base de données FrameNet contient des descriptions des cadres sémantiques et des items lexicaux en association avec leur représentation sémantique et syntaxique en contexte (la représentation des valences). Le projet inclut aussi des phrases exemples pour les frames décrits. Les semantic frames sont définis comme des représentations schématiques de situations impliquant des participants, propositions et d'autres rôles conceptuels. Prenons le frame de JUDGEMENT qui possède les rôles : judge, evaluatee, reason et le STATEMENT Frame avec des rôles comme SPEAKER, ADDRESSEE et MESSAGE, tel que démontré en (2) et en (3). FrameNet utilise aussi le concept des rôles thématiques mais qu'ils appellent des Frames Elements qui agissent comme des étiquettes pour les divers arguments des verbes. Toutefois, dans FrameNet ces éléments

sont spécifiques aux concepts du frame et non domain-independent, ainsi il en découlent de nombreux frame elements. Ça montre ici qu'on a plein de frame elements différents et ça n'apporte pas grand chose de les différencier, car en termes de fonction ils opèrent la même chose syntaxiquement. suivi d'un exemple avec tableau pour montrer la différence entre un synset X et un frame X (prendre l'exemple de l'article de FrameNet)

FrameNet (Baker, Fillmore et Lowe, 1998) : Les unités lexicales sont décrites en termes de frame semantics. Puis les unités lexicales ainsi que leurs frame semantics sont confirmées par des phrases exemples manuellement annotées sémantiquement provenant de corpus de la langue anglaise. Le but était encoder les connaissances sémantiques des unités lexicales dans une forme que les machines peuvent lire. Ils ont couvert les domaines sémantiques suivant : santé, chance, perception, communication, transaction, temps, espace, corps, motion, étapes de la vie, contexte sociaux, émotion et cognition. Ça nous donne une base de données lexicales. Qu'on décompose en 3 modules : 1 lexicon qui contient toutes les unités lexicales, une base de données de frames et les exemples annotés correspondant aux frames. La base de données de frames contient les descriptions des frames qui sont encodées en structures conceptuelles. Et les phrases exemples manuellement annotées sont des preuves empiriques que les frames ont lieu d'être. Les frames veulent décrire la structure argumentielle des unités lexicales. Ces arguments sont décrits par des étiquettes similaires aux rôles thématiques, mais beaucoup plus vastes car il y a différentes étiquettes en fonction du frame en question. On les appelle des frame elements et ceux-ci sont en lien avec le frame dont ils font partie. À noter qu'il y existe aussi une organisation hiérarchique où on a des sous-frames qui héritent de traits des frames parents. En frame semantics, un frame correspond à un scénario qui implique une interaction et des participants (ceux-ci jouent un rôle dans l'interaction)(Shi et Mihalcea, 2005)

utilisation en NLP : On a combiné VN et FN pour en faire un semantic parser.(Shi et Mihalcea, 2005). Semantic parser : Au lieu de faire un parse tree syntaxique, ça fait un parse tree sémantique et ça nous recrée un arbre en tenant compte des participants et de leur relation avec le verbe et entre eux.

mapping à VerbNet :(Shi et Mihalcea, 2005) Le mapping s'est fait en deux étapes. d'abord, il fallait mapper les entrées lexicales de VN avec des frames appropriés de FN. Ensuite, il fallait mapper les rôles sémantiques de VN avec les frame elements de FN.

2.1.3.3. XTAG

: Les chercheurs du projet XTAG ont construit une grammaire de la langue anglaise en se basant sur le formalisme de TAG (Tree Adjoining Grammar). Elle offre des descriptions riches

des verbes en anglais. Chaque item lexical se fait assigner un ensemble d'arbres décrivant ses propriétés syntaxiques de base. Les arbres reflètent la structure prédicat-argument de ces items. Les arbres peuvent se combiner en deux opérations : substitution et adjonction. En adjoignant de nouvelles branches ou en substituant des branches, la grammaire TAG permet de rendre compte des divers phénomènes linguistiques de la langue anglaise. La grammaire de XTAG inclut des descriptions syntaxiques pour 33 000 items lexicaux dont 9000 verbes et un éventail de 1300 arbres décrits et organisés en 70 famille d'arbres. (parler du mapping)

a lexicalized tree adjoining grammar for English (Research Group, 2001) Les arbres dans des familles d'arbres sont liés par le fait qu'ils représentent les mêmes type de cadres de sous-catégorisation. Ce qui distingue 2 arbres dans une même famille est des alternances syntaxiques. Les classes verbales sont organisées ainsi : Chaque verbe dans le lexicon syntaxique est associé à plusieurs familles d'arbres, où chaque famille regorge d'arbres individuels issues de différentes transformation syntaxiques mais ayant un cadre de sous-catégorisation les unissant. Ainsi, de cette manière un même verbe peut pointer vers différents arbres individuels, en passant par les familles d'arbres, et en étant associé à diverses familles d'arbres. Ainsi, ils n'ont pas à spécifier tous les arbres individuels qu'un verbe peut prendre, mais des familles d'arbres. Les familles d'arbres sont ainsi créées en prenant une forme canonique d'un cadre de sous-catégorisation et de lui faire toutes les transformations que permettent la langue anglaise, puis de générer les arbres en découlant avec les 2 opérations et stocker le tout dans la base de données d'arbres de XTAG.

XTAG system (Doran, Egedi, Hockey, Srinivas et Zaidel, 1994) Ce que fait XTAG, une phrase en input, puis on la fait passer par deux filtres : un analyseur morphologique et un POS tagger. Maintenant qu'on a des unités lexicales étiquetées et morphologiquement annotées. Ce filtrage augmente la rapidité du parsing. Puis le parser poursuit avec les unités qui lui sont fournis pour faire sa job de parser. Le parser consulte la base de données syntaxiques et la base de données d'arbres pour trouver l'arbre approprié pour chaque item lexical. Puis le parser combine les arbres élémentaires avec les 2 opérations adjonctions/substitution ce qui donne en output le parsing de la phrase donnée en input. Le travail similaire qui a été fait est la base de données d'arbres qui sont regroupés en famille d'arbres. La base de données syntaxiques pointe vers les arbres associées à chaque entrée lexicale, ou les familles d'arbres. On distingue arbre individuel (généralement non-verbaux) vs famille d'arbres (généralement des verbes) car les familles d'arbres représentent des cadres de sous-catégorisation. Les arbres dans une famille d'arbre sont liés entre eux par des transformation (changement de diathèse)

Assigning XTAG trees to VerbNet(Ryant et Kipper, 2004) Les auteurs de VerbNet ont ainsi voulu mapper leurs cadres syntaxiques aux arbres de XTAG. Ça a permis d'étendre leur

coverage à d'autres constructions syntaxiques. Comme XTAG a tenté de répertorier toutes les transformations possibles qu'un cadre de sous-catégorisation peut subir, il y avait bcp de matière. VerbNet couvre surtout la voix déclarative, tandis que XTAG explore toutes les transformations possibles. Mais ne fait pas de distinctions pour les différents sens des verbes et ne traite pas de sémantique. (juste dire qu'il y existe un mapping) et qu'ils sont allés étendre leur coverage un peu.

2.1.3.4. *Lexical conceptual structures*

La base de données de Dorr provient des théories de Jackendoff qui argumente en faveur d'une approche de décomposition sémantique pour la sémantique des verbes en termes de leur structure conceptuelle lexicale (Dorr, 1992). Un LCS c'est un graphe, c'est une représentation sémantique et le lexique, les verbes sont encodés ainsi et dans la structure conceptuelle on voit la structure syntaxique qui découle des primitifs sémantiques. Une structure est composée de : représentation hiérarchique non linéaire, il y a une tête logique (la racine du graphe), un sujet logique (un seul) , des arguments logiques et des modificateurs logiques. L'élément principal d'un LCS sont des constituents conceptuels, les primitifs conceptuels et les champs sémantiques. Les constituents conceptuels appartiennent à un ensemble de catégories : chose, évènement, état, lieu, chemin, propriété, but, manière, montant, temps. Les champs sémantiques sont des traits comme : +temp, +loc, +poss qui agissent comme des restrictions sélectionnelles. Les primitifs conceptuels : ÊTRE, ALLER, RESTER , CAUSER, INCHOATIF, EXTENSION. Une décomposition sémantique des verbes en termes de structures lexicales conceptuelles explique leur propriété syntaxiques. Un peu comme Levin l'avait perçue, les propriétés sémantiques des verbes vont influencer leur comportements syntaxiques. Avec ce formalisme, on pense que les verbes avec des LCS similaires partagent aussi des comportement syntaxiques comme des alternances de diathèses. Ils utilisent aussi des rôles thématiques pour montrer la structure argumentale. La base de données de Dorr prend aussi la hiérarchie de Levin pour structurer son information. Ainsi, les verbes sont classés en fonction des classes LCS. La base de données de Dorr LCS, inclut une grande quantité de verbes organisés en classes sémantiques dérivés des classifications de Levin.

Une structure LCS capture la sémantique d'une classe de verbe. Ainsi, tous les membres d'une classe partagent la même structure sémantique, mais c'est le contenu sémantique qui diffère à l'intérieur de cette classe pour satisfaire les contraintes lexicales des verbes. (Traum et Habash, 2000). Un graphe LCS est une représentation sémantique où il y a des noeuds dont une racine. Chaque noeud a des spécifications avec des types d'information comme : type, primitif et champ. Type; event, state, path, manner, etc. Puis après avoir spécifier le

type, on spécifie le primitif sémantique du verbe (être, aller, rester, etc.) et les champs sont des traits qui agissent comme des restrictions sur les noeuds.

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. (Schuler, 2005)

Application en NLP : A été utilisé pour faire de la traduction automatique entre des langues différentes (Dorr, 1992). Pour le projet UNITRAN, espagnol, anglais et allemand. À l'aide de représentation basées sur la LCS, ils pouvaient fournir un input sémantique qui ensuite pouvait s'appliquer aux langues traitées et générer des traductions équivalentes entre les langues en utilisant leur lexicon et la théorie LCS. Une représentation sémantique qui pourra s'appliquer aux trois langues. Et les lexicons respectifs se chargent des spécificités de chaque langues.

2.1.3.5. *Comlex*

Base de données en anglais, développée à NYU. C'est une ressource syntaxique riche, mais qui n'est pas accessible au public. Chaque item lexical a des traits syntaxiques et des compléments. Il existe 92 sous-catégorisation disponibles pour les verbes. Contient des descriptions syntaxiques pour 6000 verbes. Selon les auteurs de VerbNet, il ne fait pas de distinction entre les différents sens d'un verbe (ce qui est problématique), 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.

Le but de Comlex était de (Grishman, Macleod et Meyers, 1994) créer un dictionnaire syntaxique sur les verbes de l'anglais à des fins computationnelles. Ils ont opté pour un theory-neutral système. Surtout orienté subcategorization frame. Une entrée de dictionnaire contient : la POS, les verbes qui prennent des compléments auront un trait sous-catégorisation. Il y a aussi moyen de préciser des traits plus précis lorsqu'un verbe demande des particularités. Toutes les entrées sont faites manuellement car il ne pensait pas que des méthodes automatiques pouvait bien rendre compte des verbes moins fréquents, et ils voulaient que leurs entrées sois flawless.

2.1.3.6. *LexSchem*

A large subcategorization lexicon for french verbs (Cédric Messiant et Korhonen, 2008) L'auteur postule une automatisation de la tâche. La tâche étant la création d'un lexicon.

L'information la plus utile qu'un lexicon peut offrir, sont les cadres de sous-catégorisation des verbes (la structure prédicat-argument). Les SCF capturent au niveau syntaxique les différentes combinaisons d'arguments qu'un prédicat peut prendre. Des lexicons de sous-catégorisation peuvent être très bénéfiques à des applications NLP. Peuvent être utilisés pour le parsing, la classification sémantique, extraction d'information et traduction automatique. L'acquisition automatique a fait ses preuves dans des applications réelles, bien que contrairement à l'approche manuelle, elle n'est pas aussi accurate et détaillée. Ce qui est utile, ce sont les statistiques, la fréquence d'utilisation d'un SCF par rapport à un autre pour un verbe donné. Développé un système qui faisait une acquisition automatique des SCF en français à partir de corpus non-annoté. Les SCF acquis automatiquement sont incorporés dans lexSchem. D'abord, ils prennent des données brutes. Tag et lemmatise les mots , pour ensuite parser le tout , et commencer à en extraire les SCF. Les entrées lexicales sont composées essentiellement de : L'unité, ses cadres de sous-catégorisation et des phrases exemples tirées de corpus ainsi que la fréquence d'utilisation du SCF. (Acquis automatiquement, pourrait être intéressant pour le français à notre système et l'amener vers un GATM très riche syntaxiquement).

2.1.3.7. *A large SCF lexicon for NLP apps : Vallex*

(Korhonen, Krymolowski et Briscoe, 2006) Commence son article en justifiant pourquoi créer un lexicon verbal automatiquement est la meilleure manière de fonctionner de nos jours. Elle commence avec l'argument qu'il existe déjà des lexicons verbaux fait manuellement, mais que ceux-ci sont proies à faire des erreurs et que c'est coûteux en termes de temps à entretenir et à enrichir. Une information clé qu'on ne retrouve pas dans les dictionnaires manuellement encodés, ce sont les statistiques d'utilisation d'un SCF pour un prédicat donné. Comme beaucoup de systèmes fonctionnent avec des méthodes probabilistes, de telles informations sont cruciales à leur bon fonctionnement. Selon eux, ce genre d'information est quasi-impossible à encoder manuellement. C'est pourquoi elle postule l'acquisition des SCF automatique pour créer un dictionnaire, en extrayant les trucs via des textes non annotés. Les dictionnaires qui pratiquent cette méthode focusent que sur les verbes, ils rassemblent de l'information sur les cadres de sous-catégorisation. Mais ne différencie pas les sens des verbes, 80-85 % de token recall. Ce ne sont pas des dictionnaires hautement précis : différencie pas les sens des verbes, pas de mapping entre la représentation syntaxique et la représentation sémantique, pas toujours les alternances de diathèses.

Voici comment le lexicon est créé : Ils ont utilisé le système d’acquisition de Briscoe et Carroll (Briscoe, Carroll et Watson, 2006) capable de catégoriser 163 cadres de sous-catégorisation et retournant la fréquence des SCF. Ce système utilise la méthode RASP. Donc des données brutes sont d’abord tokénisées, étiquetées, lématisées puis parsées utilisant RASP. Puis les SCF sont extraits des phrases parsées. Puis les patrons de régime sont classés dans l’une des 163 types de SCF. Ainsi, chaque entrée lexicale est en fait une combinaison verbe/SCF ce qui donne le lexicon de base. Puis, il est filtré car, puisque c’est une méthode automatique, le système trouve des SCF qui n’en sont peut-être pas, ou bien qui sont mal formés, etc. il faut donc les retirer du dictionnaire. Il y a aussi le problème, que certains SCF ne seront pas extraits puisqu’ils ne seront pas perçus comme étant des SCF selon le système. Ainsi, il faut pallier à ces problèmes et pour ce faire ils utilisent des méthodes statistiques pour améliorer le rappel/précision. Et ils utilisent des dictionnaires construits manuellement pour trouver des SCF manquant.

Une entrée lexicale dans le dictionnaire comprend entre autre : la combinaison d’un verbe et d’un SCF, la syntaxe des arguments, la fréquence d’utilisation du SCF, les POS des tokens.

2.1.3.8. *VDE-Valency dictionary of English*

Le VDE est un dictionnaire de patron de régime/valence/cadre de sous-catégorisation, etc. Les dictionnaires de valence donne des informations sur les unités lexicales par rapport à la manière dont elles se combinent (le type d’argument que les verbes sélectionne, etc.). Ces ressources sont précieuses pour les applications NLP si on veut analyser la richesse d’une langue correctement. (Proisl et Kabashi, 2010) Le VDE contient la description de valence pour 511 verbes (et des noms et des adjectifs). Chaque entrée démontre une valence possible pour un verbe accompagné d’un exemple provenant de Bank of English. À la base le projet n’était pas destiné à une application NLP, mais les auteurs se sont rendus compte qu’il y avait un marché-là, notamment du côté de LFG qui s’y prête bien pour des applications NLP. Ainsi est né Erlangen Valency Pattern Bank qui vise surtout les constructions syntaxiques possibles. Ainsi, le projet du Pattern Bank découle du VDE qui se prête bien à des applications NLP. Le pattern bank veut donner des outils pour la recherche en linguistique sur la valence. Ça liste les patrons de valence identifiés pour les verbes par le VDE. Pour chaque patron de valence, une liste des unités lexicales pour le patron. Les 511 verbes ont été choisis sur la base qu’ils sont fréquents dans la langue anglaise, démontre des propriétés complexes et sont utiles pour des apprenants de l’anglais. Les patrons de valence qu’on retrouve dans le VDE proviennent d’une étude de corpus fait sur le COBUILD. Les patrons sont décrits en termes de syntaxe de surface. Ne tient pas compte de description sémantique. 2 verbes peuvent pointer vers les

mêmes patrons de valence s'ils ont la même structure. Dans le VDE, on donne l'information sur les participants, ce qui permet de distinguer 2 verbes différents qui partagent les mêmes patrons de valence, mais ils ne partageront pas toute la même information sémantique, c'est là qu'on précise les traits des participants. Mais, ces infos ne se retrouvent pas dans le Pattern Bank. Les patrons de valence sont identifiés par les compléments qui les remplisse. Ils distinguent les sens des lexèmes (accept au sens de approve vs integrate) et ainsi les différents patrons de valence incluent. De même que la différence des patrons entre voix passive et voix active.

Donc, couvre peu de verbes, mais pourrait extraire ce qu'ils ont et le mapper à ce qu'on a déjà. On sait que c'est du travail qui a été fait manuellement, donc devrait pas avoir de truc de fou.

2.1.3.9. *Dicovallence*

Est un dictionnaire de valence pour la langue française. Le dictionnaire comprend les cadres syntaxiques de 3700 verbes. Une entrée lexicale dans ce dictionnaire est la combinaison d'un cadre valenciel et du verbe. Ainsi, puisque la plupart des verbes comptent plus d'un cadre valenciel, il y a plus de 8000 entrées lexicales dans ce dictionnaire. Les informations à l'intérieur des cadres valenciels sont présentées dans le cadre de l'approche pronominale en syntaxe (citation). Ce que plusieurs appellent des participants, des actants, des arguments, etc. sont ici appelés des paradigmes. A été créé dans une optique de TAL et d'enseignement de la langue. Ainsi une entrée est composée du cadre valenciel en termes de paradigme. Ce qui fait que le cadre valenciel de supprimer est composé de p0 p1, une sorte de description brève de la syntaxe de l'entrée. Dans ce jargon, p0 est l'équivalent du sujet et p1 de l'objet direct, puis p2 de l'objet indirect, et les autres p sont des obliques (quantité, temps, manière, locatif, délocatif, etc.). Contient aussi des phrases exemples, les traductions du verbe en anglais et en allemand, le frame en soi avec des restrictions sélectionnelles sur les paradigmes et d'autres informations grammaticales classiques : le type d'auxiliaire et les reformulations passives possibles.

2.1.4. Utilisation de VerbNet dans des applications NLP

VerbNet a été utilisé dans un nombre impressionnant de travaux de recherche. Ce qui en témoigne de son efficacité et de sa réputation. Voici notamment quelques projets de recherche où les chercheurs se sont servis de cette base de données lexicales pour effectuer leurs travaux. Tâche : faire 3 lignes par article et décrire rapidement en quoi VerbNet a été utile pour ces travaux.

automatically building conceptual graphs using VerbNet and WordNet

Putting Pieces Together : Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing

Question Answering with Lexical Chains Propagating Verb Arguments A question answering system based on VerbNet frames

a supervised algorithm for verb desambiguasation into VerbNet classes semantic classifications for detection of verbs metaphor VerbNet class assignment as a wsd task

multilingual NLG within abstractive generation.

2.1.5. 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 prête plus. p.210 dans le livre Melcuk on pourrait garder les rôles pis les mettre dans MATE aussi

2.1.6. 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. Pour ensuite comparer avec l'utilisation que nous en aurons fait et à quel point les bons points selon eux ne sont peut-être pas les mêmes bons points que nous avons vu. Est-ce que VerbNet contient des coquilles car il a été créé par un processus automatique ? Voir la dissertation. (automatic techniques for extending coverage)

2.1.7. Pourquoi avoir choisi VerbNet ?

Expliquer pourquoi on a choisi VerbNet parmi tous les dictionnaires qui s'offraient à nous (en excluant ceux dans d'autres langues). D'autres ont aussi utilisé VerbNet pour ce genre de tâche, notamment : jonathan pfeil et Offer Biller

le tutoriel : ESCW tuto RelExtra v1

On est dans la même optique que Wanner et Mille qui pensent que des ressources comme VerbNet pourraient aussi être bénéfique pour la NLG, malgré le fait qu'elles sont plus conçues pour d'autres tâches.

2.2. PYTHON

À l'aide du module *xml.etree.cElementTree* nous avons pu faire des opérations sur l'ensemble des données de VerbNet qui sont encodées dans des fichiers XML. Le module *Etree* nous permet de naviguer dans les fichiers XML de VerbNet puis de manipuler et d'extraire les données qui nous intéressent. Après avoir manipulé et extrait les données intéressantes, nous les avons compilées dans des dictionnaires. Ceux-ci seront utilisés par notre système MATE pour générer du texte automatiquement. Par la suite, nous avons aussi utilisé Python et le module *Etree* pour extraire les phrases exemples qui accompagnent chaque patron de régime couvert par la ressource lexicale. Dans le but de créer des structures sémantiques qui nous serviront d'input pour générer en syntaxe de surface la phrase exemple et ainsi vérifier si MATE est capable de générer les phrases exemples.

Nous voulions donc créer des documents DICT qui seraient l'information lexicale de notre système de GAT. Il est d'abord important de dire dans les grandes lignes comment notre système MATE fonctionne. Ainsi, notre système prend en input une représentation sémantique d'un énoncé, puis fait appel à des dictionnaires et des règles de grammaire pour générer en output une représentation syntaxique de surface de l'énoncé. Les unités lexicales s'encodent d'une certaine manière dans MATE et nous voulions créer nos dictionnaires à l'image des contraintes données par le système MATE. Nous avons donc codé la manipulation et l'extraction des données en fonction de la manière dont le lexique est encodé dans MATE.

2.2.1. Création du dictionnaire de verbes : `lexicon.dict`

La première étape de notre projet consistait à créer le dictionnaire verbal. Bien que le dictionnaire final comporte 6393 entrées lexicales verbales, cette partie de code nous permettait de créer la base de notre système. Il serait important de décrire dans les grandes lignes à quoi ressemble le dictionnaire pour mieux expliquer pourquoi nous l'avons extrait de cette manière. Ainsi MATE possède la caractéristique d'héritage des traits. Donc, en ce qui concerne les verbes, nous avons architecturé la chose de cette manière. Il existe une classe des verbes qui ont les traits suivants : une `dpos=V` et une `spos=verb`.

```
verb {  
    dpos = V  
    spos = verb  
}
```

Ensuite, les 278 classes verbales de VerbNet pointent vers la classe `verb`, donc les classes héritent de ces deux traits. Ce qui fait en sorte qu'on n'a pas à répéter dans chaque entrée de classe verbale les traits `dpos` et `spos`.

```

"absorb-39.8": verb {
  gp = { id=NP_V_NP                      dia=12 } // Cotton absorbs water.
  gp = { id=NP_V_NP_PP_from_source      dia=123 } // Cattle take in nutrients from their fe
}

```

Ensuite, chaque membre de chaque classe verbale pointe vers la classe ou la sous-classe qui le représente ce qui fait en sorte qu'il hérite de tous les traits de la classe(sous-classe) vers laquelle il pointe. Par exemple : absorb, ingest, take in vont tous trois hériter des traits de l'entrée "absorb-39.8" qui est la classe verbale. C'est de cette manière que notre dictionnaire passe de 278 entrées lexicales à 6393. Car, les verbes de la langue anglaise ont été classés dans des classes verbales et nous les intégrons à notre dictionnaire en gardant la même architecture que VerbNet avait pensé (basé sur les travaux de Levin).

```

absorb : "absorb-39.8"
take_in : "absorb-39.8"
ingest_1 : "absorb-39.8"

```

2.2.1.1. *Création du dictionnaire initial contenant les 278 classes verbales comme entrées lexicales*

LISTING 2.6. code pour lexicon.dict

```

# BLOC 1
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

# BLOC 2
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 frame.findall('SYNTAX/PREP')+frame.findall('SYNTAX/LEX')]
        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(preps[2]))

    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

# BLOC 3
with open('lexicon.dict','w') as f:
    f.write('lexicon {\n')
    for file in [f for f in os.listdir('verbnets') if f[-4:] == '.xml']:
        root = ET.parse('verbnets/'+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 } // ' + ' '.join(gp[1])) for gp in d[c]]
            f.write('\n}\n')
    f.write('\n}')

```

(Peut-être rajouter les numéros de lignes, ce sera utile pour faire référence à des lignes de code) Ce premier script Python se découpe en trois blocs. Les deux premiers blocs sont des fonctions que nous définissons pour ensuite les utiliser dans le BLOC 3 qui nous permettra d'écrire le dictionnaire initial. Ainsi, pour une meilleure compréhension du script complet, nous décrirons les blocs 1 et 2 pour expliquer ce que les fonctions font, ensuite nous décrirons le bloc 3 car l'usage des fonctions sera maintenant clair.

Commençons par la première fonction `supers`. La fonction prend 2 arguments : `t` et `i`. Si vous regardez le bloc 3, dans la section `sc`, on voit que `supers` prend les arguments (`root`, `'verb'`). Ces deux arguments sont les suivants : `root` donne accès à l'arbre XML et `'verb'` est fait pour que chaque classe verbale pointe vers la catégorie verbe. Pour en hériter les traits `dpos=V` et `spos=verb`. On va chercher le trait ID dans l'un des deux arguments (l'argument arbre donc montré dans listing le trait ID dans le XML). Ensuite on crée un dictionnaire s'appelant `sc` dont la clé est le ID et la valeur est `i`. Puis on instancie une variable `subclasses` qui contient les racines de chaque sous-classes dans les documents XML (montrer avec listings

à quoi ça ressemble dans le document XML). Puis ensuite, pour chaque élément dans sous-classe (si élément il y a) on lui passe la fonction supers et on va chercher son identifiant car en lui repassant la fonction, on retourne a sub.get('ID') donc on va chercher le ID qui se retrouve dans la partie sous-classe de l'arbre de la classe verbale et on met à jour le dictionnaire sc en y ajoutant les identifiants des sous-classes (les clés) qui pointeront vers l'identifiant de la classe qui le gouverne, car le deuxième argument de supers est une chaîne de caractère, dans ce cas, la chaîne ID qu'on avait pu chercher plus tôt dans la fonction. Cela est fait dans le but de construire : sous-classe : classe (donc la sous-classe hérite des traits de sa classe) grâce au mécanisme d'héritage de MATE, et tel que VerbNet le mentionne, chaque sous-classe hérite des traits donnés par la classe qui la domine.

Maintenant, passons à la seconde fonction treeframe. Cette fonction prend un argument. On va aller chercher ID. Puis on initialise une liste z. Pour tous les frames qu'on retrouve dans l'arbre en question, on va chercher la description primaire du frame. On va ensuite se servir des expressions régulières pour uniformiser les descriptions à notre bon vouloir. Parallèlement, nous avons aussi créé une liste s'appelant exclude qui contient toutes les descriptions que nous voulons exclure de notre système et si une description dans un frame se retrouve aussi dans exclude, alors on n'y touche pas et elle ne sera pas modifiée ni ajoutée à la liste description. Ensuite, on modifie encore le nom des descriptions car nous ferons une opération par la suite. Donc, on modifie toutes les descriptions qui contiennent PP pour qu'elles ressemblent à

2.2.1.2. *Extraction des 6393 membres des classes verbales pour enrichir le dictionnaire lexicon.dict*

LISTING 2.7. 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, je 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')
```

2.2.2. Création du gpcon

LISTING 2.8. 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],

# CREATION 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 valeurs dans mon dictionnaire de gpcon.

2.2.3. Scripts pour faire les tests

2.2.3.1. *Extraction des exemples*

LISTING 2.9. 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('verbnnet') if f[-4:] == '.xml']:
        root = ET.parse('verbnnet/'+file).getroot()
        d = (treeframes(root))
        final_liste = liste + d
        [f.write(x+'\n') for x in final_liste]
```

2.2.3.2. *Création des structures qui serviront de tests*

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

LISTING 2.10. 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
f.write(structure)
g.write(structure)

```

Chapitre 3

INTÉGRATION DES RÉGIMES DE VERBNET DANS GENDR

3.1. COMMENT GÉNÉRER DU TEXTE AVEC GENDR

3.1.1. Dictionnaires

3.1.2. Grammaires

3.1.3. Graphes

3.2. IMPLÉMENTATION DES PATRONS DE RÉGIME

3.2.0.1. *GP dans les dictionnaires*

3.2.0.2. *GP dans les règles grammaticales*

3.3. MÉCANISME DSYNT=CREATED->CONSTRAINED->OK POUR GÉNÉRER UN ÉNONCÉ SIMPLE (SUJET, VERBE, OBJET)

3.3.1. 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.

3.3.2. 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.

3.3.3. 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 actanciellles 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.

3.3.4. application des règles actanciellles

Une fois qu'un `gp` est sélectionné, on appliquera la règle actancielle qui lui correspond. Nos règles actanciellles 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 soustraire 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]

3.3.5. 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 actanciellles 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 conviennent 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.

3.3.6. 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 règles de types *guess* opèrent lorsqu'il nous manque certaines infos.

3.3.7. `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 à un noeud déjà existant. Ce noeud existe déjà car il a soit été créé par `root_standard` ou par une des règles actanciellles. 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 le 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 lexicalisés, sinon, l'arbre sera incomplet et la génération échouera.

3.3.8. 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 passage 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

Chapitre 4

ÉVALUATION DU SYSTÈME

(PG:) gauche
(PD:) droite
(COND:) conditions

FIGURE 4.1. lexstd

4.1. MÉTHODES D'ÉVALUATION EN NLG

notre système ne passera pas par la phase de morpho et linéarisation. Mais, avec la structure de surface, nous pouvons déjà savoir si l'arbre en output est grammaticalement correcte selon les arbres de dépendance. Les outils de linéarisation ne font que prendre les arbres et linéariser en fonction des dépendances.

voir : The first Surface Realisation Shared task

l'évaluation de notre système sera principalement humaine, mais notre manière de voir si c'était bon sera de comparer les structures de surface avec les phrases données en input provenant de VerbNet et voir si ça a de l'allure

évaluation manuelle : avec les jpg, la version textuelle des graphes, et les phrases exemples

notre évaluation concernera : les limites de GenDR et les limites de VerbNet, une critique théorique de ce qui font, en testant une application pratique de leur lexicon. Et fournir une ligne directrice aux futurs lexicons, ou bien aux genres de lexicon qu'il nous faut. Dans le fond, une critique de Levin aussi, qui ne traite pas des fonctions lexicales, car c'est une analyse syntaxique qui sous-tend des principes sémantiques. Et la partie sémantique de VerbNet est plus axé sur les rôles sémantique et sur les prédicats sémantiques, mais qui clash avec notre théorie qui rend mieux compte de l'interface sémantique/syntaxe.

dire que c'est normal que en NLG, les méthodes d'évaluation sont complexes. Surtout que notre truc est pas linéarisé, pourquoi on peut pas utiliser de F-mesure, etc.

Bibliographie

Ayan, N. F. et B. J. Dorr. 2002, «Generating A Parsing Lexicon from an LCS-Based Lexicon», dans *In : Proceedings of the LREC-2002 Workshop on Linguistic Knowledge Acquisition and Representation, Las Palmas, Canary Islands*, p. 4352.

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.

Briscoe, T., J. Carroll et R. Watson. 2006, «The Second Release of the RASP System», dans *Proceedings of the COLING/ACL on Interactive Presentation Sessions, COLING-ACL '06*, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 77–80, doi :10.3115/1225403.1225423.

Cédric Messiant, T. P. et A. Korhonen. 2008, «LexSchem : A Large Subcategorization Lexicon for French Verbs», dans *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, édité par Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Daniel Tapias, European Language Resources Association (ELRA), Marrakech, Morocco, ISBN 2-9517408-4-0. [Http ://www.lrec-conf.org/proceedings/lrec2008/](http://www.lrec-conf.org/proceedings/lrec2008/).

Copestake, A. 1992, «The ACQUILEX LKB : Representation issues in semi-automatic acquisition of large lexicons», dans *Proceedings of the 3rd Conference on Applied Natural Language Processing (ANLP-92)*, p. 88–96.

Doran, C., D. Egedi, B. A. Hockey, B. Srinivas et M. Zaidel. 1994, «XTAG System : A Wide Coverage Grammar for English», dans *Proceedings of the 15th Conference on Computational Linguistics - Volume 2, COLING '94*, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 922–928, doi :10.3115/991250.991297.

Dorr, B. J. 1992, «The Use of Lexical Semantics in Interlingual Machine Translation», *Machine Translation*, vol. 7, n° 3, p. 135–193, ISSN 0922-6567.

Fellbaum, C., éd.. 1998, «WordNet : An Electronic Lexical Database», Language, Speech, and Communication, MIT Press, Cambridge, MA, ISBN 978-0-262-06197-1.

Fillmore, C. J. 1968, «The Case for Case», dans *Universals in Linguistic Theory*, édité par E. Bach et R. T. Harms, Holt, Rinehart and Winston, New York, p. 0–88.

- Grishman, R., C. Macleod et A. Meyers. 1994, «Complex 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.
- Jackendoff, R. 1972, «Semantic Interpretation in Generative Grammar», Cambridge : Mass., Mit Press.
- 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>).
- Korhonen, A., Y. Krymolowski et T. Briscoe. 2006, «A large subcategorization lexicon for natural language processing applications», dans *In Proceedings of LREC*.
- Lareau, F., F. Lambrey, I. Dubinskaite, D. Galarreta-Piquette et M. Nejat. 2018, «GenDR : A Generic Deep Realizer with Complex Lexicalization», dans *Proceedings of 11th Edition of the Language Resources and Evaluation Conference (LREC)*, Miyazaki.
- Levin, B. 1993, «English verb classes and alternations : A preliminary investigation», .
- Palmer, M., D. Gildea et P. Kingsbury. 2005, «The Proposition Bank : An Annotated Corpus of Semantic Roles», *Comput. Linguist.*, vol. 31, n° 1, doi :10.1162/0891201053630264, p. 71–106, ISSN 0891-2017.
- Proisl, T. et B. Kabashi. 2010, «Using High-Quality Resources in NLP : The Valency Dictionary of English as a Resource for Left-Associative Grammars», dans *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, édité par N. C. C. Chair), K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner et D. Tapias, European Language Resources Association (ELRA), Valletta, Malta, ISBN 2-9517408-6-7.
- Research Group, T. X. 2001, «A Lexicalized Tree Adjoining Grammar for English», *IRCS Technical Reports Series*.
- Ryant, N. et K. Kipper. 2004, «Assigning XTAG Trees to VerbNet», dans *Proceedings of the 7th International Workshop on Tree Adjoining Grammar and Related Formalisms*, Vancouver, Canada, p. 194–198.
- Schuler, K. K. 2005, «Verbnet : A Broad-coverage, Comprehensive Verb Lexicon», PhD Thesis, University of Pennsylvania, Philadelphia, PA, USA.

Shi, L. et R. Mihalcea. 2005, «Putting Pieces Together : Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing», dans *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing'05, Springer-Verlag, Mexico City, Mexico, ISBN 3-540-24523-5 978-3-540-24523-0, p. 100–111, doi :10.1007/978-3-540-30586-6_9.

Traum, D. et N. Habash. 2000, «Generation from Lexical Conceptual Structures», dans *Proceedings of the 2000 NAACL-ANLP Workshop on Applied Interlinguas : Practical Applications of Interlingual Approaches to NLP - Volume 2*, NAACL-ANLP-Interlinguas '00, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 52–59, doi :10.3115/1117554.1117561.

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 renseigne-

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

ments, consultez le site [web de la FESP](#).

Annexe B

LE TITRE2

texte annexe B

