Structured prediction methods for semantic parsing

Alban Petit

université PARIS-SACLAY

Structured prediction methods for semantic parsing

Méthodes de prédiction structurée pour l'analyse sémantique

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat: Informatique
Graduate School : Informatique et Sciences du Numérique.
Référent : Faculté des Sciences d'Orsay.

Thèse préparée dans l'unité de recherche **Laboratoire Interdisciplinaire des Sciences du Numérique** (Université Paris-Saclay, CNRS), sous la direction de **François YVON**, Directeur de recherche, et le co-encadrement de **Caio CORRO**, Maître de conférences

Thèse soutenue à Paris-Saclay, le 9 février 2024, par

Alban PETIT

Composition du jury

Membres du jury avec voix délibérative

Vincent GUIGUE Professeur, Université Paris-Saclay	Président
Maxime AMBLARD Professeur des universités, LORIA / CNRS, Université de Lorraine	Rapporteur & Examinateur
Marie CANDITO Maîtresse de conférences HDR, LLF, Université Paris-Cité	Rapporteur & Examinatrice
Pascal DENIS Chargé de recherche, INRIA / CNRS, Université de Lille	Examinateur
Laura KALLMEYER Professeure, Heinrich Heine University Düsseldorf	Examinatrice

THESE DE DOCTORAT
NNT : 2024UPASG002

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Méthodes de prédiction structurée pour l'analyse sémantique

Mots clés: analyse sémantique, généralisation compositionnelle, traitement automatique des langues

Résumé: L'analyse sémantique est une tâche qui consiste à produire une représentation formelle manipulable par un ordinateur à partir d'un énoncé en language naturel. Il s'agit d'une tâche majeure dans le traitement automatique des langues avec plusieurs applications comme le développement de systèmes de question-réponse ou la génération de code entre autres. Ces dernières années, les approches fondées sur les réseaux de neurones, et en particulier les architectures séquence-à-séquence, ont démontré de très bonnes performances pour cette tâche. Cependant, plusieurs travaux ont mis en avant les limites de ces analyseurs sémantiques sur des exemples hors distribution. En particulier, ils échouent lorsque la généralisation compositionnelle est requise. Il est donc essentiel de développer des analyseurs sémantiques qui possèdent de meilleures capacités de composition. La représentation du contenu sémantique est une autre préoccupation lorsque l'on aborde l'analyse sémantique. Comme différentes structures syntaxiques peuvent être utilisées pour représenter le même contenu sémantique, il est souhaitable d'utiliser des structures qui peuvent à la fois représenter précisément le contenu sémantique et s'ancrer facilement sur le langage naturel. À ces égards, cette thèse utilise des représentations fondées sur les graphes pour l'analyse sémantique et se concentre sur deux tâches. La première concerne l'entrainement des analyseurs sémantiques fondés sur les graphes. Ils doivent apprendre une correspondance entre les différentes parties du graphe sémantique et l'énoncé en langage naturel. Comme cette information est généralement absente des données d'apprentissage, nous proposons des algorithmes d'apprentissage qui traitent cette correspondance comme une variable latente. La deuxième tâche se concentre sur l'amélioration des capacités de composition des analyseurs sémantiques fondés sur les graphes dans deux contextes différents. Notons que dans la prédiction de graphes, la méthode traditionnelle consiste à prédire d'abord les nœuds, puis les arcs du graphe. Dans le premier contexte, nous supposons que les graphes à prédire sont nécessairement des arborescences et nous proposons un algorithme d'optimisation basé sur le lissage des contraintes et la méthode du gradient conditionnel qui permet de prédire l'ensemble du graphe de manière jointe. Dans le second contexte, nous ne faisons aucune hypothèse quant à la nature des graphes sémantiques. Dans ce cas, nous proposons d'introduire une étape intermédiaire de superétiquetage dans l'algorithme d'inférence. Celle-ci va imposer des contraintes supplémentaires sur l'étape de prédiction des arcs. Dans les deux cas, nos contributions peuvent être vues comme l'introduction de contraintes locales supplémentaires pour garantir la validité de la prédiction globale. Expérimentalement, nos contributions améliorent de manière significative les capacités de composition des analyseurs sémantiques fondés sur les graphes et surpassent les approches comparables sur plusieurs jeux de données conçus pour évaluer la généralisation compositionnelle.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title: Structured prediction methods for semantic parsing

Keywords: semantic parsing, compositional generalization, natural language processing

Abstract: Semantic parsing is the task of mapping a natural language utterance into a formal representation that can be manipulated by a computer program. It is a major task in Natural Language Processing with several applications, including the development of questions answers systems or code generation among others. In recent years, neural-based approaches and particularly sequence-to-sequence architectures have demonstrated strong performances on this task. However, several works have put forward the limitations of neural-based parsers on out-of-distribution examples. In particular, they fail when compositional generalization is required. It is thus essential to develop parsers that exhibit better compositional abilities. The representation of the semantic content is another concern when tackling semantic parsing. As different syntactic structures can be used to represent the same semantic content, one should focus on structures that can both accurately represent the semantic content and align well with natural language. In that regard, this thesis relies on graph-based representations for semantic parsing and focuses on two tasks. The first one deals with the training of graph-based semantic parsers. They need to learn a correspondence between the parts of the semantic graph and the natural language utterance. As this information is usually absent in the training data, we propose training algorithms that treat this correspondence as a latent variable. The second task focuses on improving the compositional abilities of graph-based semantic parsers in two different settings. Note that in graph prediction, the traditional pipeline is to first predict the nodes and then the arcs of the graph. In the first setting, we assume that the graphs that must be predicted are trees and propose an optimization algorithm based on constraint smoothing and conditional gradient that allows to predict the entire graph jointly. In the second setting, we do not make any assumption regarding the nature of the semantic graphs. In that case, we propose to introduce an intermediate supertagging step in the inference pipeline that constrains the arc prediction step. In both settings, our contributions can be viewed as introducing additional local constraints to ensure the well-formedness the overall prediction. Experimentally, our contributions significantly improve the compositional abilities of graph-based semantic parsers and outperform comparable baselines on several datasets designed to evaluate compositional generalization.

Remerciements

Tout d'abord, je souhaite remercier mes encadrants de thèse François Yvon et Caio Corro sans qui je ne serai pas arrivé au bout de ces trois années de doctorat. Merci pour votre disponibilité, votre implication dans mes travaux, la qualité de votre encadrement ainsi que la patience dont vous avez sû faire preuve y compris quand la direction de recherche à suivre n'était pas forcément claire.

Je remercie ensuite Maxime Amblard, Marie Candito, Pascal Denis, Vincent Guigue et Laura Kallmeyer qui m'ont fait l'honneur d'être les membres de mon jury. Merci pour l'attention que vous avez porté à mes travaux ainsi que pour vos commentaires pertinents dans les rapports et lors de la soutenance.

Je souhaite ensuite remercier toutes les personnes au LISN et dans le département STL qui ont contribué à rendre ces années agréables. Merci à Aina, Alina, Aman, Anh Khoa, Dávid, Elise, Francesca, Hugues, Jitao, Léa-Marie, Léo, Marc, Maxime, Minh Quang, Paul, Rémi, Simon, Sofiya, Syrielle, Théo, Tom et Yajing pour les moments partagés au LISN ou en-dehors. Je remercie François de m'avoir accueilli comme voisin de bureau pendant deux ans, j'ai apprécié de le partager avec toi. Je tiens à remercier particulièrement Shu pour ces années de thèse, notamment pour les souvenirs de conférence. Je garderai des souvenirs de cette semaine d'aventures à Dublin pendant très longtemps.

Je tiens aussi à remercier mes amis pour le soutien ou les moments de détente que vous m'avez apporté pendant ces années : Adrien, Anne-Laure, Bartho, Egon, Joé, Nicolas, Marion, Rémi, Sid, Tonin et Yves. Je remercie particulièrement Julien pour ces (trop rares) sorties à vélo ensemble et Mehdi pour ces soirées de détente ou de discussions profondes sur le canapé.

Je voudrais remercier ma famille pour les encouragements qu'ils m'ont toujours té-

moigné. Tout particulièrement, je remercie mes parents pour avoir soutenu mes choix tout au long de mon parcours et avoir toujours été présents quand j'avais besoin d'aide ou de réconfort.

Finalement, je veux remercier Natalia : tu as été là pour moi, particulièrement sur cette dernière année de thèse où j'avais parfois l'impression que je n'arriverai pas au bout de la rédaction. Tu as fait preuve d'une patience infinie envers moi et je n'y serai pas arrivé sans toi.

Contents

List of Acronyms

CCG Combinatory categorical grammar.

GVCNNSA Generalized valency-constrained not-necessarily spanning arborescence.

ILP Integer Linear Programming.

LMO Linear minimization oracle.

LSE LogSumExp.

MAP Maximum a posteriori.

MCD Maximum Compound Divergence.

MGVCNNSA Maximum generalized valency-constrained not-necessarily spanning arborescence.

MR Meaning representation.

MSCG Maximum Spanning Connected SubGraph.

NLP Natural Language Processing.

RV Random variable.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

1.1.1 Semantic parsing

Language is at the center of human communication as it allows us to understand and exchange with others. Notably, humans are able to compose words into sentences, understand the meaning of a sentence and perceive the different meanings of an utterance based on its context. On the other hand, computers do not possess this ability. By design, they can only operate on languages like programming or query languages that possess a strict grammar that must be followed. Otherwise, the computer will be unable to execute the utterance. Additionally, the meaning of each element in a formal language must be precisely defined. This makes communication between humans and computers difficult: the human is required to use a language that can be understood by the computer, which is not possible for a large majority of the population, and even in that case, communication is limited to the domain of the language used.

As such, one long standing goal of natural language processing has been the development of systems that convert natural language to a representation that can be manipulated by a computer. Such systems enable a variety of tasks in human-computer interaction: question answering, information extraction, instruction interpretation, code generation, etc ... The task of mapping a natural language utterance into a formal meaning representation is known as **semantic parsing**. We give several examples of natural

Sentence	How big is Texas?
MR	`answer(A, (size(B, A), const(B, stateid(texas))))`
Sentence	The cat ate the cake.
MR	`*cat(x_2) ; *cake(x_5); ate.agent(x_3, x_2) AND ate.theme(x_3, x_5)`
Sentence	The dog ate the bone.
MR	`(e / eat-01 :ARG0 (d / dog) :ARG1 (b / bone))`

Table 1.1: Examples of natural languages utterances paired with their meaning representations (MR). The first representation is a logical form following the Prolog syntax taken from the GeoQuery (Zelle and Mooney, 1996) dataset. The second representation is a logical form taken from the COGS (Kim and Linzen, 2020) dataset. The third one is the Penman notation (Kasper, 1989) of the Abstract Meaning Representation (Banarescu et al., 2013).

language utterances with corresponding meaning representations in Table 1.1.

Early approaches for semantic parsing were rule-based and focused on very limited domains (Woods et al., 1972; Winograd, 1972). As natural language can be flexible and ambiguous, these parsers required significant manual work to be developed and did not generalize to other domains. To reduce the amount of work required, ulterior research focused on the development of semantic parsers that could be trained from pairs consisting of an utterance and its corresponding semantic representation. Various research directions were explored throughout the years: grammar-based approaches (Zelle and Mooney, 1996; Kate et al., 2005; Zettlemoyer and Collins, 2005, 2007; Kwiatkowski et al., 2011), sequence-to-sequence approaches (Jia and Liang, 2016; Dong and Lapata, 2016; Iyer et al., 2017; Wang et al., 2020), span-based approaches (Pasupat et al., 2019; Herzig and Berant, 2021) or graph-based approaches (Flanigan et al., 2014; Zhou et al., 2016; Lyu and Titov, 2018) among others.

These approaches tackle various semantic structures. However, the structures of the natural language utterance and the semantic representation do not necessarily match. Rambow (2010) argued that different syntactic structures could represent the same semantic content. It is thus essential to focus on structures that can both represent accurately semantic content and align fairly well with natural language. In that regard, we focused on graph-based representations in this thesis.

1.1.2 Compositionality of language

The principle of compositionality states that

> "The meaning of an expression is a function of the meanings of its parts and of the way they are syntactically combined." (Frege, 1956; Partee, 1984)

Indeed, humans can understand the meaning of natural language thanks to systematic compositionality, the algebraic capacity to understand and produce novel combinations from known components (Chomsky, 1957; Montague, 1970). Montague argued that natural language could be mapped to logical forms via a grammar, *i.e.* the semantic content would be obtained by combining adjacent elements together. This idea was adopted in early semantic parsing systems as they relied on hand-crafted, and later learned grammars. While these systems were limited, they had an explicit inductive bias for compositionality.

The development of sequence-to-sequence architectures was an important step in semantic parsing. These approaches cast semantic parsing as a string-to-string machine translation problem where the semantic representation was linearized and treated as a foreign language. While these approaches performed well, semantic parsing tasks designed to evaluate the robustness to **compositional generalization** were proposed to study whether they had an inductive bias (Lake and Baroni, 2018). Their work as well as further contributions (Loula et al., 2018; Finegan-Dollak et al., 2018) demonstrated that sequence-to-sequence approaches fail to learn compositional generalization.

Therefore, recent work in semantic parsing has focused on the development of approaches with an inductive bias for compositional generalization. In this thesis, we propose two novel approaches that improve the compositional abilities of graph-based semantic parsers. While the assumptions made for each approach are different, our key idea is the same in both cases: we ensure the well-formedness of the global prediction by introducing additional local constraints.

1.2 Contributions

The contributions of this thesis can be summed up as follows:

- We highlight how graph-based methods allow for a unified approach that can be applied to a variety of domains and meaning representations. The only requirement is a bijective mapping between the target representation and a graph representation.

- Graph-based parsers must learn a correspondence between parts of the formal representation and the sentence. Explicitly annotating this correspondence in the data can be problematic: how ambiguities are dealt with, which correspondences are allowed (one-to-one, one-to-many, many-to-one, ...), how phenomenons like reentrancies are handled, etc ... As such, this information is usually absent. We propose training algorithms that treat this correspondence as a latent variable, a setting that we refer to as "weakly-supervised training".

- In graph prediction, the traditional pipeline is to first predict the nodes and then the arcs of the graph. This can lead to additional errors as the prediction of nodes is not informed by the prediction of arcs. In the case where the structures that must be predicted are trees, we propose to predict the entire graph jointly. Our inference algorithm solves the linear relaxation to this problem and delivers an optimality certificate when the solution returned is integral. In addition, when given a semantic grammar, our algorithm guarantees that its output will be well-formed with respect to the grammar.

- The arcs of the graph are usually predicted independently from each other. It has been observed that graph-based parsers fail to predict them correctly when they are required to generalize to unseen sentence structures. To tackle this issue, we get rid of the independence assumption. We propose to introduce an intermediate supertagging step in the inference pipeline. In this step, each node is assigned a supertag that will constrain the arcs that can be predicted as adjacent to the node in the arc prediction step.

1.3 Outline

Based on our contributions, this thesis is organized as follows.

Chapter 2. Semantic parsing and structured prediction. In this chapter, we introduce the semantic parsing task. First, we present the major families of semantic formalisms. Then, we describe the main research directions over the years for semantic parsing with their motivations, advantages and limitations. Notably, we discuss the recent focus on neural-based parsers and improving their compositional abilities. After that, we give a brief overview of related works for compositional generalization in semantic parsing. Finally, we discuss the evaluation of semantic parsers.

Chapter 3. Graph-based semantic parsing. The following chapter introduces graph-based approaches for semantic parsing. We first motivate the use of these approaches before presenting them, the issues that they raise and how they are handled. Then, we formally define the task as it is currently tackled and present common neural architectures. After that, we present the graph-based approaches that were developped concurrently to our contributions for compositional generalization. Finally, we introduce the datasets that are used in this thesis and how they are transformed into graphs.

Chapter 4. Combinatorial optimization and training. In this chapter, we study the training of a graph-based parser in a setting where the anchoring of a semantic graph over its corresponding sentence is not available. We propose an objective function for this setting and prove that its computation requires to solve a NP-hard problem. We introduce a combinatorial optimization framework and propose a new algorithm to solve the linear relaxation of this problem and, ultimately, compute the objective function.

Chapters 5 and 6. Improving compositional generalization. These chapters present two novel graph-based approaches, each with their advantages and shortcomings. In Chapter 5, we focus on semantic representations without reentrancies, *i.e.* their graph representations are trees, and for which typed grammars are available. For this setting,

we propose an approach that does not restrict the search space and covers all the structures observed in practice, including non-projective ones. Our proposed approach predicts the entire graph jointly while ensuring the well-formedness of the predicted solution with respect to the semantic grammar.

In Chapter 6, we do not impose a priori constraints on the semantic representation. We study a generalization task that requires semantic parsers to handle both known grammatical structures with words that were not observed in these structures before as well as novel grammatical structures. As semantic parsers tend to struggle particularly on the latter, we propose a novel multi-step inference that relies on supertagging to predict the semantic graph. Notably, our supertagging step ensures that there exists a feasible solution.

Chapter 7. Conclusion. Finally, we conclude the thesis with a summary of our contributions. We also provide an overview of potential future research directions.

1.4 List of publications

The articles listed below were published during the thesis.

Auto-encodeurs variationnels : contrecarrer le problème de posterior collapse grâce à la régularisation du décodeur
Authors: Alban Petit, Caio Corro
Conference: 28e Conférence sur le Traitement Automatique des Langues Naturelles
Publication date: June 2021
Language: French
Abstract: Variational autoencoders are generative models useful to learn latent representations. In practice, on text generation tasks, they tend to ignore latent variables during the decoding process. We propose a new regularization method based on “fraternal” dropout to encourage the use of these latent variables. We evaluate our approach on multiple datasets and observe improvements in all the tested configurations.

Preventing posterior collapse in variational autoencoders for text generation via decoder regularization

Authors: Alban Petit, Caio Corro

Workshop: Deep generative Models and Downstream Applications Workshop at NeurIPS 2021

Publication date: December 2021

Language: English

Abstract: Variational autoencoders trained to minimize the reconstruction error are sensitive to the posterior collapse problem, that is the proposal posterior distribution is always equal to the prior. We propose a novel regularization method based on fraternal dropout to prevent posterior collapse. We evaluate our approach using several metrics and observe improvements in all the tested configurations.

Un algorithme d'analyse sémantique fondée sur les graphes via le problème de l'arborescence généralisée couvrante

Authors: Alban Petit, Caio Corro

Conference: 29e Conférence sur le Traitement Automatique des Langues Naturelles

Publication date: June 2022

Language: French

Abstract: We propose a novel algorithm for graph-based semantic parsing via the maximum generalized spanning arborescence problem.

On Graph-based Reentrancy-free Semantic Parsing

Authors: Alban Petit, Caio Corro

Journal: Transactions of the Association for Computational Linguistics, Volume 11

Publication date: 2023

Language: English

Abstract: We propose a novel graph-based approach for semantic parsing that resolves two problems observed in the literature: (1) seq2seq models fail on compositional generalization tasks; (2) previous work using phrase structure parsers cannot cover all the semantic parses observed in treebanks. We prove that both MAP inference and latent tag anchoring (required for weakly-supervised learning) are NP-hard problems. We propose two optimization algorithms based on constraint smoothing and conditional gradient to approximately solve these inference problems. Experimentally, our approach delivers state-of-the-art results on GeoQuery, Scan, and Clevr, both for i.i.d. splits and for splits that test for compositional generalization.

Structural generalization in COGS: Supertagging is (almost) all you need

Authors: Alban Petit, Caio Corro, François Yvon

Conference: The 2023 Conference on Empirical Methods in Natural Language Processing

Publication date: December 2023

Language: English

Abstract: In many Natural Language Processing applications, neural networks have been found to fail to generalize on out-of-distribution examples. In particular, several recent semantic parsing datasets have put forward important limitations of neural networks in cases where compositional generalization is required. In this work, we extend a neural graph-based parsing framework in several ways to alleviate this issue, notably: (1) the introduction of a supertagging step with valency constraints, expressed as an integer linear program; (2) the reduction of the graph prediction problem to the maximum matching problem; (3) the design of an incremental early-stopping training strategy to prevent overfitting. Experimentally, our approach significantly improves results on examples that require structural generalization in the COGS dataset, a known challenging benchmark for compositional generalization. Overall, these results confirm that structural constraints are important for generalization in semantic parsing.

Chapter 2

Semantic parsing and structured prediction

This chapter introduces the necessary context for readers to understand the contributions presented in this thesis. We first present the major families of semantic formalisms in Section 2.1. Then, we give an overview of the first manually designed systems for semantic parsing and the development of grammar-based parsing in Section 2.2. Section 2.3 focuses on neural-based approaches. Sequence-to-sequence architectures are presented first with their limitations with respect to compositional generalization. Then, we introduce span-based approaches which draw inspiration from previous grammar-based approaches to improve compositional generalization. In Section 2.4, we give a brief summary of related works that study compositional generalization for semantic parsing. Finally, we discuss the evaluation of semantic parsers in Section 2.5.

2.1 Semantic formalisms

The role of a semantic representation is to provide a structured representation of the meaning of a natural language utterance. To do so, it requires an ontology rich enough to represent the concepts, their properties and relations for a given task. In this section, we present the most common formalisms used in semantic parsing.

Sentence	How many states border texas ?
Lambda calculus	`count(`λ`x.state(x)`$\wedge$` borders(x, texas))`
Prolog (Zelle and Mooney, 1996)	`answer(A, count(B, (state(B), next_to(B,C), const(C, stateid(texas))), A))`
λ-DCS (Liang, 2013)	`count(Type.USState`$\sqcap$`Borders.Texas)`
FunQL (Kate et al., 2005)	`count(state(next_to_2(stateid(texas))))`

Table 2.1: Comparison of multiple semantic representations that can equivalently be reduced to first-order logic augmented with lambda calculus (Carpenter, 1997).

2.1.1 Logical forms

The modelisation of human language has a long history that precedes the task of semantic parsing. As presented by Mooney (2014), existing work on this topic dates back to at least 1685. Gottfried Leibniz argued that if a language could be evaluated, it could be possible to determine whether a reasoning was correct or not. To do so, he developed the *characteristica universalis*, a formal conceptual language that could be evaluated by the *calculus ratiocinator*, an automated reasoner. This work was a precursor to the Boolean algebra (Boole, 1854) which reduces propositional logic to an algebra over binary variables with the conjunction ($\wedge$), disjunction ($\vee$) and negation ($\neg$) operators. The Boolean algebra is however limited. Whitehead and Russell (1912) finalized the development of the modern first-order predicate logic. It introduces quantified variables with the $\forall$ and $\exists$ operators as well as predicates and functions. While first-order logic is unable to manipulate sets, Carpenter (1997) proposed to augment it with lambda calculus. This formalism is widely used in semantic parsing. Note that works may rely on different representations that can equivalently be reduced to first-order logic augmented with lambda calculus, see Table 2.1.

2.1.2 Graph-based representations

While logical forms are popular formalisms in semantic parsing, there has also been a growing interest in the development of general-purpose meaning representations based on graphs. Representing the semantic content as a labeled graph where vertices denote entities and edges represent the semantic relations between them offers several advantages: they can easily model semantic phenomenons like co-references and are fairly

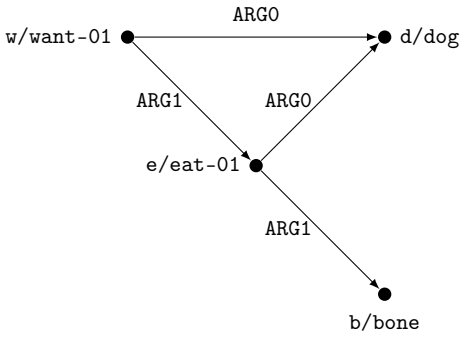

Figure 2.1: The semantic graph corresponding to "The dog wants to eat a bone" in the AMR formalism.

intuitive to understand for a human. In addition, as graph theory has been extensively studied in computer science, there exists a large variety of algorithms that can be leveraged to train a parser predicting graphs.

As graph-based representations grew in popularity in recent years, several formalisms have been proposed. We can cite Abstract Meaning Representation (AMR, Banarescu et al., 2013), Universal Conceptual Cognitive Annotation (UCCA, Abend and Rappoport, 2013), Universal Dependency Semantics (UDS, White et al., 2016) or more recently, BabelNet Meaning Representation (BMR, Martínez Lorenzo et al., 2022), among others. As these representations are not the focus of this thesis, we refer to Abend and Rappoport (2017) for a survey of these representations and their semantic differences. We give an example of a graph-based semantic representation in Figure 2.1.

2.2 Development of grammar-based parsing

2.2.1 Rule-based semantic parsers

A detailed survey of early semantic parsing systems can be found in Androutsopoulos et al. (1995). In this section, we present how these systems operated and their limitations. Montague (1970) argued that natural language could be mapped to logical forms by applying recursive syntactic rules. Early semantic parsers explored this direction and relied on a hand-crafted grammar to recursively parse natural language utterances and map them to logical forms. The first step in these systems was to produce a tree representation of the sentence using a grammar. The latter could either be syntax-based, *e.g.*

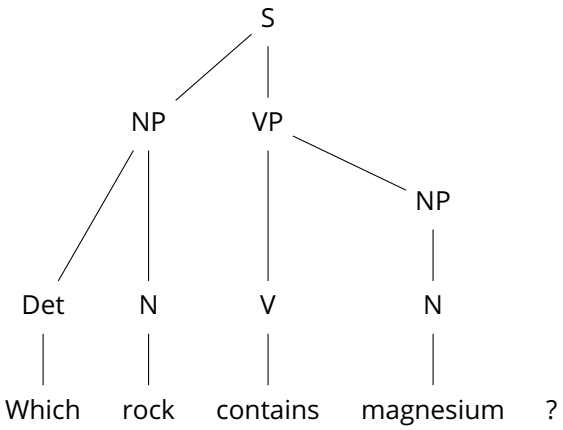

Figure 2.2: Syntactic parse of the query "Which rock contains magnesium ?" using the grammar defined in Example 2.1.

in the LUNAR system (Woods et al., 1972), or semantic-based (Lockemann and Thompson, 1969; Waltz, 1978; Hendrix et al., 1978; Templeton and Burger, 1983).

Example 2.1: Parsing of an utterance

We illustrate the parsing step with a query that could be asked to the LUNAR system: “Which rock contains magnesium ?”. We use a simplified syntax-based grammar defined as follows:

- S $\rightarrow$ NP VP
- NP $\rightarrow$ Det N | N
- Det $\rightarrow$ “which” | “what”
- N $\rightarrow$ “rock” | “magnesium”
- VP $\rightarrow$ V NP
- V $\rightarrow$ “contains” | “emits”

This grammar indicates that a simple declarative clause (S) is composed of a noun phrase (NP) and a verb phrase (VP), a noun phrase is composed of a determiner (Det) and a noun (N), a noun can either be “rock” or “magnesium”, etc ... Using this grammar, the query will be parsed into the tree shown in Figure 2.2.

The second step consisted in mapping the parse tree obtained into a logical form. To

do so, each element appearing in the grammar was assigned a semantic rule defining how it should be mapped. The mapping of non-terminal elements in the grammar is usually defined recursively. Thus, the parse tree is traversed bottom-up as shown in Example 2.2.

Example 2.2: Mapping of a parse tree to a logical form

We use the same query and grammar as in Example 2.1. We assume that the semantic rules are defined as follows:

- "which" is mapped to `for_every X`.
- "rock" is mapped to `(is_rock X)`.
- "contains" is mapped to `contains`.
- "magnesium" is mapped to `magnesium`.
- The mapping of a NP is the mapping of its determiner (if applicable) and the mapping of its noun.
- The mapping of a VP is `(`V' `X` NP'`)` where V' and NP' are the mappings of its verb and noun phrase respectively.
- The mapping of a S is `(`NP' VP'`; (printout X))` where NP' and VP' are the mappings of its NP and VP respectively.

By following these rules, each word is first mapped to its representation. Then, the NP is mapped to `for_every X (is_rock X)`. The VP is mapped to `(contains X magnesium)` and finally, the S is mapped to `(for_every X (is_rock X) (contains X magnesium); (printout X))`.

As the grammar was hand-crafted, these systems were limited and they could not handle out-of-vocabulary words or unknown grammatical structures. In addition, the semantic rules could not be transferred to other domains, meaning that they needed to be redesigned from scratch for every system. Due to these issues, the focus of research shifted from rule-based methods to statistical approaches relying on data.

2.2.2 Statistical learning

Statistical approaches for semantic parsing relied on pairs composed of a sentence and its corresponding semantic representation. As the downside of rule-based parsers was the hand-crafted grammar, the aim of these approaches was to infer a grammar from the examples provided in the training set. An early example is the CHILL system (Zelle and Mooney, 1996) that infers a deterministic shift-reduce parser. To evaluate their system, Zelle and Mooney (1996) built a corpus from sentences submitted by uninformed users. On these sentences, their approach outperformed an existing rule-based system known as Geobase. This system was improved later on with a refined algorithm to infer the rules of the shift-reduce parser (Tang and Mooney, 2001; Thompson and Mooney, 2003).

The next evolution came with the introduction of Combinatory Categorical Grammars (CCG, Steedman, 1996, 2000) for semantic parsing. In semantic parsing, a CCG is generally defined by a lexicon in which an entry is a pair composed of a word and an associated category containing both syntactic and semantic information. In addition, a CCG also has a set of combinators, *i.e.* rules indicating how the syntactic and semantic informations of adjacent elements in the utterance can be composed together. Zettlemoyer and Collins (2005) proposed to reduce semantic parsing for logical forms to learning a probabilistic CCG, *i.e.* a CCG where each possible derivation is given a weight. Given pairs of sentences with their corresponding logical forms, they aim to find the weights that maximizes their joint likelihood. As polynomial-time algorithms to parse CCGs (Vijay-Shanker and Weir, 1993; Kuhlmann and Satta, 2014) exist, predicting the logical form associated to a natural language utterance is reduced to computing the derivation of maximum weight via dynamic programming. Probabilistic CCGs proved to be a popular research direction for semantic parsing in the following years. Zettlemoyer and Collins (2007) introduced additional combinators to handle variable word orderings or missing words. Kwiatkowski et al. (2011) improved the induction of the lexicon by introducing factored lexicons, *i.e.* words that belong to the same class (*e.g.* "Texas" and "California" are both states) are grouped in the same template.

Word	Syntactic information	Semantic information
Utah	NP	$utah$
Idaho	NP	$idaho$
borders	$(S\backslash NP)/NP$	$\lambda x.\lambda y.borders(y,x)$

Table 2.2: Example of a simple CCG lexicon containing 3 entries.

Utah	borders	Idaho
NP	$(S\backslash NP)/NP$	NP
$utah$	$\lambda x.\lambda y.borders(y,x)$	$idaho$
	$S\backslash NP$	
	$\lambda y.borders(y,idaho)$	
	S	
	$borders(utah,idaho)$	

Figure 2.3: Parsing of the sentence "Utah borders Idaho" with the CCG defined in Example 2.3.

Example 2.3: Parsing with a CCG

We illustrate parsing with a CCG on the sentence "Utah borders Idaho". The lexicon of the CCG is given in Table 2.2. We assume that there exists two combinators:

- $A/B : f \quad B : g \quad \Rightarrow \quad A : f(g)$
- $B : g \quad A\backslash B : f \quad \Rightarrow \quad A : f(g)$

The first combinator means that two elements with the syntactic types A/B and B can be combined to produce an element with the type A. The second one means that the types B and $A\backslash B$ can also be combined to produce an element with the type A. In both cases, the semantic information of the resulting element will be the application of f to g. Given this CCG, the resulting parse is shown in Figure 2.3.

Note that other research directions were also explored. An example is training semantic parsers with less supervision to bypass the need for complex annotations in datasets (Berant et al., 2013; Kwiatkowski et al., 2013; Pasupat and Liang, 2015). As these topics are not the focus on this thesis, we do not discuss them here.

2.3 Neural-based semantic parsing

The development of neural networks led to new research directions being explored for semantic parsing. Most of them fall under the paradigm of structured prediction.

> **Definition 2.4: Structured representation**
>
> A structured representation $z \in \mathcal{Z}$ is a discrete object consisting of interdependent parts $p \in \mathcal{P}$ respecting consistency constraints (Niculae et al., 2023).

> **Definition 2.5: Structured prediction**
>
> Given an input x, structured prediction consists in finding a structured representation z of maximum weight inside a set $\mathcal{Z}^x$ with respect to a weighting function $f^x : \mathcal{Z}^x \to \mathbb{R}$. The set of possible representations $\mathcal{Z}^x$ is usually specific for each input x and has an exponential size, *i.e.* it is impossible to enumerate every single element in $\mathcal{Z}^x$. Additionally, the parts of a structured representation are highly constrained.

Based on Definition 2.5, one of the main concerns when tackling semantic parsing is how the problem should be cast and which structure should be used to model the semantic representation. In this section, we present the major approaches for neural semantic parsing and discuss their compositional abilities as well as computational implications.

2.3.1 Sequence-to-sequence architectures

Presentation. Wong and Mooney (2006) and Andreas et al. (2013) proposed to rely on machine translation methods for semantic parsing. The approach proposed by Andreas et al. (2013) was to linearize the semantic representation and treat it as a foreign language. Machine translation saw significant improvements with the development of sequence-to-sequence architectures (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015). It was thus appealing to also use them for standard structured prediction problems (Vinyals et al., 2015). As these models do not make any independence assumption, the likelihood of an output sequence y given an input sentence w can be expressed using the chain rule

as follows:

$$p(y_1, ..., y_n|w) = p(y_1|w)\ \ p(y_2|y_1, w) \ ... \ p(y_n|y_1, ..., y_{n-1}, w)$$
$$= \prod_{i=1}^{n} p(y_i|y_{<i}, w)$$

As there is an exponential number of possible sequences, finding the most likely sequence is intractable. The decoding algorithm in these models usually relies on beam search, *i.e.* the tokens are generated iteratively and the k likeliest structures are kept in the beam, where k is a constant.

This approach quickly gained traction in semantic parsing (Jia and Liang, 2016; Dong and Lapata, 2016; Iyer et al., 2017) as it outperformed previous work and was straightforward. Multiple variants for sequence-to-sequence were proposed to improve their accuracy. Dong and Lapata (2016) proposed a tree decoder to better model the structure of the semantic representation. Their LSTM-based decoder could produce a special token indicating a sub-tree. The hidden state at that point is then given to two separate decoders: one predicting the sequence that would be inserted in place of the special token and another predicting the rest of the semantic representation. Dong and Lapata (2018) proposed a two-step encoding-decoding process: the first step would predict a sketch of the semantic representation from the sentence. The final semantic representation would then be predicted from this sketch. Zheng and Lapata (2021) introduced an intermediate tagging step where each word is associated a semantic symbol representing its meaning. The sequence-to-sequence model then predicts a semantic representation conditioned on the sentence and the semantic tags. Reordering the words in the sentence has also been explored by Wang et al. (2020) and Lindemann et al. (2023). In each case, the idea is to provide additional information to the sequence-to-sequence model.

Compositional generalization. As grammar-based approaches explicitly relied on rules that combine adjacent elements in the utterance, they could generalize to unseen structures as long as these structures could be parsed by the grammar. Lake and Baroni (2018) introduced the SCAN dataset to study whether such an inductive bias was present or not in sequence-to-sequence architectures. The task associated to the SCAN dataset re-

Sentence	Turn around left after jump twice
Action sequence	`I_JUMP I_JUMP I_TURN_LEFT I_TURN_LEFT I_TURN_LEFT I_TURN_LEFT`
Sentence	Look thrice and run opposite left
Action sequence	`I_LOOK I_LOOK I_LOOK I_TURN_LEFT I_TURN_LEFT I_RUN`

Table 2.3: Examples of sentences in the SCAN dataset with their corresponding action sequences.

quires the model to map a natural language command to a sequence of actions, as illustrated in Table 2.3. The experiments conducted by Lake and Baroni (2018) highlighted that sequence-to-sequence architectures failed catastrophically in two situations:

- When the sentences in the test set were mapped to longer sequences of actions than the ones seen during training.
- When a primitive, *i.e.* a command that corresponds to a basic action, seen in isolation during training is introduced in more complex structures during testing.

Further work (Loula et al., 2018; Finegan-Dollak et al., 2018; Keysers et al., 2020; Yao and Koller, 2022) reached the same conclusions.

2.3.2 Span-based approaches

The failure of sequence-to-sequence architectures for compositional generalization motivated the developement of neural-based approaches inspired by traditional grammar-based approaches. Span-based approaches were originally explored for syntactic parsing (Stern et al., 2017; Corro, 2020) and applied to semantic parsing (Pasupat et al., 2019; Herzig and Berant, 2021; Liu et al., 2021). They consist in predicting partial semantic representations over short fragments of the utterance and composing them to build the representation for the entire utterance. The idea behind this research direction is to take advantage of modern neural architectures and introduce an inductive bias that should encourage compositional generalization. In this subsection, we present two span-based approaches designed to this end: SpanBasedSP (Herzig and Berant, 2021) and LeAR (Liu et al., 2021).

SpanBasedSP. The aim of SpanBasedSP is to produce a tree where each node covers a span (i, j), *i.e.* the sequence of words from w_i to w_j, and is tagged either with a concept

from the semantic domain, a `join` operator or the empty operator ϕ. The `join` operator indicates that the meaning of a node is derived from the meaning of its children. We denote $\mathcal{C}$ the set composed of every concept as well as `join` and ϕ. The model relies on a `BERT` encoder (Devlin et al., 2019) followed by an MLP to compute a score $s(i, j, c)$ for each span (i, j) and element $c \in \mathcal{C}$. As they assume that the prediction for each span can be made independently, given θ the parameters of the neural network, the likelihood of tagging a span (i, j) with c is:

$$p_\theta(T[i, j] = c|w) = \frac{\exp s(i, j, c)}{\sum_{c' \in \mathcal{C}} \exp s(i, j, c')}$$

and the log-likelihood of a tree T is:

$$\log p(T|w) = \sum_{i<j} \log p_\theta(T[i, j] = c|w)$$

During inference, the tree T of maximum score can be found using the CKY algorithm (Cocke, 1970; Kasami, 1965; Younger, 1967). However, Herzig and Berant (2021) require that the tree obtained must be valid with respect to the semantic domain. To do so, they propose a variant of the CKY algorithm where the top k solutions are computed for each span. Then, they iterate over the top k overall trees in descending score order and return the first valid tree. We give an example of span tree produced by SpanBasedSP in Figure 2.4

LeAR. A two-step approach has been proposed in LeAR (Liu et al., 2021) to produce a semantic tree: first, a binary tree is constructed over the sentence. Then, each terminal node in this tree is tagged with a concept and each non-terminal node is tagged with a semantic operation. The binary tree is constructed by relying on a Tree-LSTM encoder (Tai et al., 2015) as follows:

- Given a sentence $w = (w_1, \ldots, w_n)$, each word w_i originally corresponds to a node v_i^1 and is represented by a vector $\boldsymbol{r}_i^1$.
- At each time step t, a score is computed for each pair of adjacent nodes. The vectors corresponding to the pair with the highest score are given to the Tree-LSTM encoder.

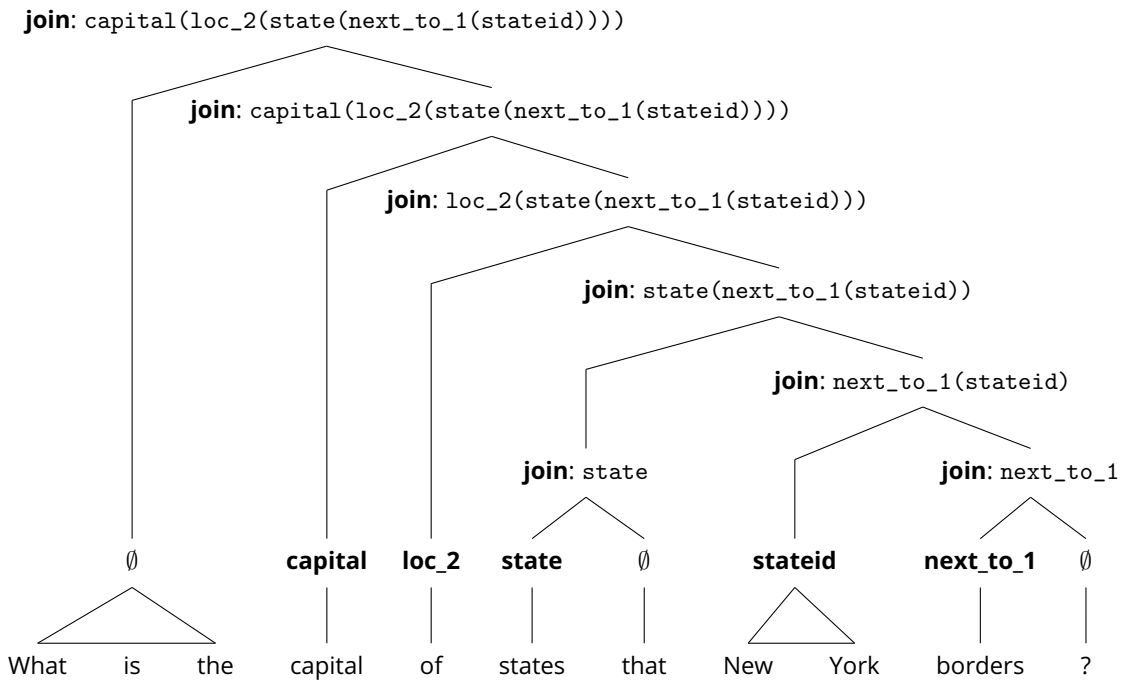

Figure 2.4: Example of a span tree that can be produced by SpanBasedSP. The prediction for each span is indicated in bold. We abbreviate "`stateid(new york)`" to "`stateid`" for clarity.

> A new node is created in the place of the pair and its representation is the output of the encoder $\boldsymbol{r}_i^{t+1}$.

When only one node remains, the binary tree can be recovered from the successive merges. We illustrate the two steps used by LeAR in Figure 2.5.

Search space limitations. Both approaches produce phrase structure trees. It is known that the complexity of inference on these structures is directly impacted by the considered search space (Kallmeyer, 2010). Importantly, (ill-nested) discontinuous phrase structure parsing is known to be NP-hard, even with a bounded block-degree (Satta, 1992).

Herzig and Berant (2021) only explore two restricted inference algorithms, both of which have a cubic time complexity with respect to the input length (Corro, 2020). The first one only considers continuous phrase structures, *i.e.* derived trees that could have been generated by a context-free grammar, and the second one also considers a specific type of discontinuities, see Corro (2020, Section 3.6). This restriction is problematic as both algorithms fail to cover the full set of phrase structures observed in the datasets

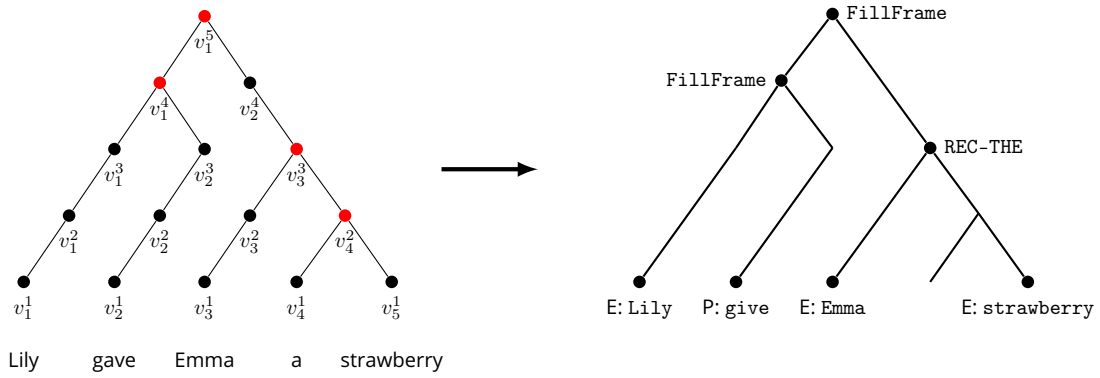

Figure 2.5: Example of a span tree that can be produced by LeAR for the sentence "Lily gave Emma a strawberry". On the left, a binary tree is constructed by iteratively merging nodes. We highlight merges in red. Once this tree is constructed, each node can potentially be tagged with a semantic primitive or operation. We represent these tags on the right. The resulting tree must then be interpreted to produce the semantic representation "`strawberry(x_5) AND give.agent(x_2, Lily) AND give.recipient(x_2, Emma) AND give.theme(x_2, x_5)`".

their approach was evaluated on. As this issue motivates the contribution presented in Chapter 5, we discuss it in more details in Section 5.1.

Liu et al. (2021) only consider continuous phrases structures as their model constructs binary trees. To fully cover the structures observed in the datasets used in their experiments, they design the set of semantic operations that are required for each domain. Requiring manual work from a human expert for each domain is highly impractical. In addition, introducing domain knowledge via these operations could leak useful information for compositional generalization to the model, thus preventing a fair evaluation.

Compositional generalization. Both models demonstrated significantly better performances compared to sequence-to-sequence architectures when compositional generalization is required. As they serve as baselines for our contributions, detailed results about them are presented in Sections 5.4 and 6.5.

2.4 Related works for compositional generalization

The recent focus on compositional generalization in semantic parsing was not limited to the development of novel structured prediction approaches. In this section, we give a brief overview of other research directions that were pursued in that regard.

2.4.1 Datasets for compositional generalization

In this subsection, we present the various methods that were used to design new or adapt existing datasets to better assess compositional generalization. Note that while we briefly present the methods here, we will provide a detailed presentation of the datasets used in our work in Section 3.5.

Alternative data split. Finegan-Dollak et al. (2018) observed that several text-to-SQL datasets tended to split examples between the training and the test set based on the natural language utterance, *i.e.* two examples must appear in the same set if their natural languages utterances only differ by their constants. However, as natural language is ambiguous, this led to examples in the training and the test set with identical SQL queries. Instead, they proposed to split the examples based on the query, *i.e.* two examples must appear in the same set if their SQL queries only differ by their constants.

Additional test set. Clevr (Johnson et al., 2017) is a dataset designed for image-based question answering that has also been adapted for semantic parsing. The questions are generated from templates with empty slots where each slot has a possible set of values. In the original dataset, the training and the test sets are generated from the same set of templates. Bahdanau et al. (2019) proposed a systematic generalization test set for Clevr known as Closure. To do so, they designed 7 new templates that are used exclusively to generate examples for Closure.

Original datasets. Novel datasets designed explicitly to evaluate compositional generalization like COGS (Kim and Linzen, 2020) and CFQ (Keysers et al., 2020) were proposed. In the case of COGS, the training data was generated using a Probabilistic Context-Free Grammar. Then, various generalization cases were identified to compose the generalization set. For each of them, examples were generated from a new PCFG. As such, the generalizations that are expected in COGS are known and allow for a straightforward error analysis.

The CFQ dataset consists of natural language questions paired with the corresponding SPARQL query. With CFQ, Keysers et al. (2020) proposed a new method known as

the Distribution-Based Compositionality Assessment to generate data splits that assess compositional generalization. It iteratively selects examples either for the training set or the test set in a way such that the distributions over atoms, *i.e.* the smallest units that compose the natural language utterance, in the two sets are as similar as possible while the distributions over compounds, *i.e.* combinations of atoms, are as different as possible. The intuition behind this method is that systematic compositionality is required to successfully parse the test set.

2.4.2 Other research directions

Data augmentation. Data augmentation has been explored with the aim of providing a compositional inductive bias to semantic parsers. Several works rely on infering the structure of the data: Jia and Liang (2016) infer a grammar that is used to generate additional data, (Qiu et al., 2022) rely on a grammar-based generative model and Yang et al. (2022) recombine sub-trees in the span-based representation proposed by Herzig and Berant (2021). A more straightforward approach proposed by Andreas (2020) identifies fragments that are used in a similar context and constructs new examples by swapping them.

Meta-learning. The impact of meta-learning on compositional generalization has been evaluated in various settings. For each batch in the training set, Conklin et al. (2021) proposed to construct a "meta-test" composed of sentences with similar atoms but different compounds. Given model parameters θ, a gradient descent step is performed with respect to the training batch to obtain parameters θ'. The training objective is then the maximization of the likelihood of the training batch given θ and the "meta-test" given θ'. The intuition behind this approach is that the parameter update on the training batch should also be beneficial for the "meta-test". Zhu et al. (2021) has shown that training a parser on multiple datasets could improve compositional generalization on an additional unseen dataset.

Auxiliary tasks. Introducing auxiliary tasks has been shown to be beneficial for compositional generalization in some cases. Jiang and Bansal (2021) observed significant im-

provements on the SCAN dataset by introducing additional sequences that needed to be predicted by the model. In addition to a word-level attention mechanism, Yin et al. (2021) added an attention loss at the span-level and observed slight improvements for recursive queries.

2.5 Evaluation

To compare multiple semantic parsers, we require metrics that can assess their respective performance. In this section, we present the two most common ways that are used to evaluate whether the semantic representation predicted by the model is correct or not. Then, we present corner cases for both approaches and highlight why it is difficult to design a unique metric for this task.

2.5.1 Match accuracy

The most straightforward way to evaluate whether a prediction is correct is by checking if it is identical to the expected output or not. The metric can either be computed over the entire semantic representation or over individual components. In the first case, it is a binary metric known as the *exact match accuracy*. This is the most common way to evaluate the match accuracy of a prediction. If the semantic representation can be decomposed into individual components, it is also possible to compute the F1-score based on the individual components of the prediction and the expected representation.

When the semantic representation contains variables, it is necessary to find a matching between the variables in the prediction and the ones in the expected representation that maximizes the metric used as using different variable names should not change the performance of a parser. Finding the matching that maximizes the F1-score is NP-hard, thus approximated algorithms like Smatch (Cai and Knight, 2013) have been proposed. On the other hand, computing the exact match accuracy can often be done efficiently.

2.5.2 Denotation accuracy

The result of the execution or evaluation of a semantic representation is known as its denotation. In tasks like database querying, it is common to have access to an executor,

i.e. a software that can execute the semantic representation against a database. A metric that is used to evaluate the performance of a semantic parser is the denotation accuracy. It is a binary metric indicating whether the answer returned by the executor matches the expected answer.

2.5.3 Corner cases

Exact match accuracy. There is a downside to using the exact match accuracy: the handling of spurious representations, *i.e.* representations that are unnecessarily more complex but carry the same meaning with respect to the domain. For example, in the FunQL formalism for GeoQuery (Kate et al., 2005), the query "Give me the state of California" can correspond to the semantic representation "`answer(stateid(california))`". In FunQL, the predicate `state` takes a set of elements as an argument and returns the subset of the elements that are states. Thus, "`answer(state(stateid(california)))`" is also a query yielding the state of California. One could even argue over which representation should be considered the "correct" one: should "state" be explicitly mentioned in the semantic representation or is its implication by "`stateid`" enough? When using exact match accuracy, only one answer is considered as correct. This can lead to considering some predictions as wrong even though they carry the same meaning.

Denotation accuracy. As only the result of the query's evaluation is considered, this metric successfully handles the corner case presented for exact match accuracy. However, it can be affected by another issue: a prediction that is obviously wrong to the human eye can be considered as correct if its execution yields the same answer. Still in the GeoQuery domain, there are no states that border Hawaii. Similarly, there are no rivers that traverse both Florida and Alaska. It is trivial for a human to see that both of these queries do not carry the same meaning. However, their denotations are equivalent and thus, they would be considered as identical when computing the denotation accuracy.

As one can see, the exact match accuracy can be too strict as it discards queries that could be argued to be correct while the denotation accuracy can be too lenient as an unrelated query can yield the same denotation and be considered as correct. Thus, to properly assess the performance of semantic parsers, one should report both metrics

when it is relevant and possible.

2.6 Conclusion

In this chapter, we introduced the major semantic formalisms and gave an overview of the main approaches to semantic parsing over the years. We highlighted that sequence-to-sequence architectures were straightforward approaches that performed well on semantic parsing tasks. However, they failed on tasks that require compositional generalization.

Span-based approaches perform better on tasks evaluating compositional generalization. This suggests explicitly anchoring fragments of the semantic representation on the sentence and composing them together introduces an inductive bias that is desirable for compositional generalization. The downside of these approaches is that they either fail to cover every phrase structure or require significant manual work to adapt the semantic representation of each domain.

It is desirable to rely on an approach that presents a similar inductive bias but without the limitations of span-based parsers during inference. In the following chapter, we present the approach that will serve as a basis for our contributions: graph-based parsing.

Chapter 3

Graph-based semantic parsing

In Chapter 2, we introduced several formal representations and presented multiple approaches to tackle semantic parsing. We postponed to this section the presentation of the approach our contributions are based on: graph-based semantic parsing.

In Section 3.1, we first present the reasons that motivated the development of graph-based methods for semantic parsing. We highlight the issues raised by these methods and how they were tackled in Section 3.2. Then, we explain how graph-based methods currently approach semantic parsing by giving a formal definition of the problem and describing the general structure of modern neural architectures in Section 3.3. The main focus of this thesis is the use of graph-based methods to improve compositional generalization. We present works developed concurrently to our contributions in Section 3.4. Finally, Section 3.5 is dedicated to presenting the datasets used in our contributions.

3.1 Motivation

Graphs are ubiquitous in computer science and provide a representation that is both simple to visualize and intuitive to understand for many linguistic entities. They have been studied extensively and there exists a large variety of algorithms to manipulate them. It is thus appealing to reduce a problem to a known graph problem as it allows to rely on existing algorithms. In addition, graphs are a convenient structure for optimization (Martins et al., 2010; Corro et al., 2017). This approach has been observed in NLP before, for dependency parsing (McDonald et al., 2005). These authors proposed to reduce weighted

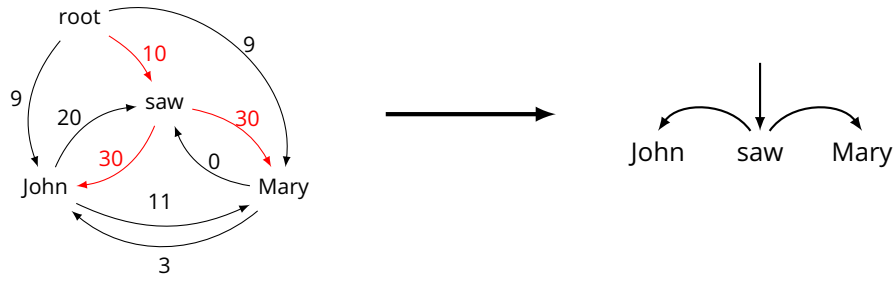

Figure 3.1: Graph-based dependency parsing for the sentence "John saw Mary". Each arc in the complete graph on the left is weighted with the score of the corresponding dependency in the sentence. The maximum spanning arborescence is represented in red. The resulting dependency parse of the sentence is illustrated on the right. This figure is inspired by McDonald et al. (2005).

dependency parsing to computing the maximum spanning arborescence in a directed graph, see Figure 3.1.

By doing so, they do not restrict the search space like other existing work (Eisner, 1997; Gómez-Rodríguez et al., 2011; Pitler et al., 2012, 2013) and can rely on the Chu-Liu/Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967), an efficient algorithm with a $\mathcal{O}(n^2)$ time complexity (Tarjan, 1977).

In Section 2.1, we introduced graph-based meaning representation formalisms like AMR (Banarescu et al., 2013). Various research directions were explored to parse these representations. A popular approach was to linearize the semantic representation and rely on sequence-to-sequence architectures (Section 2.3). However, it is appealing to draw inspiration from dependency parsing and frame the prediction of these meaning representations as a graph problem. In addition, the conversion of other formal representations like logical forms or query languages into graphs can be done in some cases with minimum manual work (see Section 3.5 for several illustrations). Thus, graph-based parsers also present the advantage of being applicable to these other formalisms and domains without requiring a large amount of work.

3.2 Semantic parsing as a graph prediction problem

In graph-based semantic representations, the semantic domain is composed of a set of concepts E and a set of relation labels L. Graph-based parsing consists in predicting a

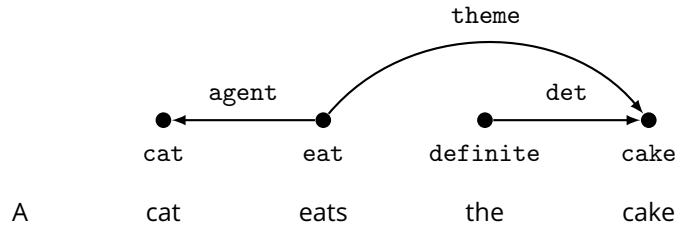

Figure 3.2: The semantic graph corresponding to the sentence "A cat eats the cake" in the COGS dataset.

semantic graph $\langle V, A \rangle$ where each vertex $v \in V$ is tagged with a concept $e \in E$ and each arc $a \in A$ is labeled with a label $l \in L$. We illustrate this in Figure 3.2.

A common assumption is that each word in the sentence can produce at most one vertex and that there is at most one arc between two vertices. If the problem was unconstrained, it would be easy to solve: one could simply select the concept and label of maximum weight (if it is positive) for each vertex and arc respectively. However, the semantic representation usually introduces hard structural constraints which prevents the existence of efficient inference algorithms. Thus, inference algorithms are usually heavily dependent on the desired output space.

One of the earliest graph-based approaches for semantic parsing was proposed by Flanigan et al. (2014). To ensure that the predicted graph is weakly connected, they rely on a two-step inference: concepts are first predicted via a semi-Markov model. Then, they construct a complete graph in which each predicted concept is a vertex and each arc is weighted with respect to a vector of manually defined features. Arcs are then selected by computing the maximum connected spanning subgraph (MSCG). Their approach was originally designed to parse AMR semantic graphs. However, in this formalism, a graph is always directed and acyclic, a constraint that is not enforced by their approach. Enforcing it would require to compute the directed acyclic subgraph of maximum weight, a problem known to be NP-hard (Kuhlmann and Jonsson, 2015; Schluter, 2015).

It was highlighted by Zhou et al. (2016) that predicting a concept from an ambiguous word without relying on the context, *i.e.* the surrounding words in the sentence, could lead to errors in the pipeline. For example, the word "fall" can either be a noun representing the season of the year or a verb representing the action of falling. To handle this, they proposed to predict the concepts and relations between them jointly by relying on an

approximate incremental algorithm.

Graph-based approaches are also influenced by the weighting functions. In that regard, semantic parsing drew inspiration from advances in neural architectures for dependency parsing. In the latter task, it was proposed to obtain contextual feature representations for each word from bi-LSTMs (Kiperwasser and Goldberg, 2016). This approach allows both the computation of weights for each word by taking the surrounding context into account and independent predictions for each word during inference. With biaffine attention (Dozat and Manning, 2017), this yielded significant improvements in dependency parsing. It was quickly adapted for semantic parsing (Dozat and Manning, 2018) and was shown to outperform existing parsers on AMR parsing (Lyu and Titov, 2018). As we will explain in Section 3.3, this neural architecture is commonly used in semantic parsing nowadays.

The main takeaway from this section is that while graph-based approaches share common principles, the inference algorithm is heavily dependent on the constraints that are imposed on the semantic representation. In addition, exact inference can become NP-hard depending on this set of constraints. We will prove in Chapters 5 and 6 that exact inference in both of our settings is NP-hard.

3.3 Formal definition and common architectures

Recent graph-based parsers (Lyu and Titov, 2018; Jambor and Bahdanau, 2022; Weißenhorn et al., 2022) as well as the contributions presented in Chapters 5 and 6 use a similar formulation of graph-based semantic parsing and rely on neural architectures that follow a common structure. We introduce both in this section.

3.3.1 Formal definition of graph-based semantic parsing

Given a sentence $w = (w_1, \ldots, w_n)$, the objective of graph-based semantic parsing is to find the semantic graph that most likely corresponds to it by determining for each word w_i whether it produces a vertex v_i and its associated concept $e \in E$ and for each pair of words $\langle w_i, w_j \rangle$ whether there is an arc between their respective vertices v_i and v_j and its associated label $l \in L$. If we introduce a special concept $\emptyset$ and a special relation $\emptyset$

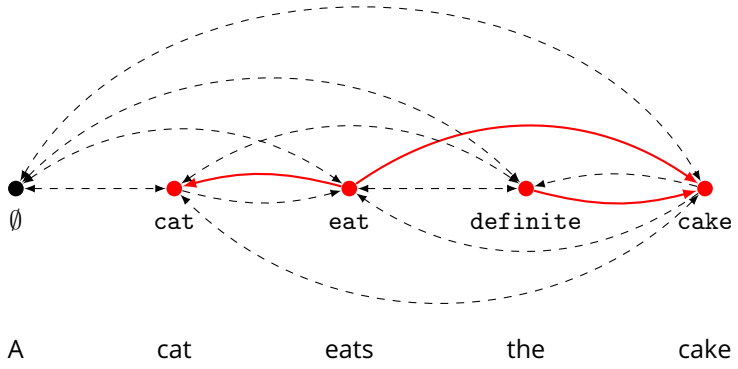

Figure 3.3: The complete graph corresponding to the semantic graph presented in Figure 3.2. The original semantic graph in highlighted in red. For clarity, we do not represent arc labels. Dashed arcs are labeled with the $\emptyset$ label.

that represent the lack of a concept or a relation respectively, this can be equivalently viewed as constructing a complete graph $G = \langle V, A \rangle$ over w, tagging each vertex $v \in V$ and labeling each arc $a \in A$ (potentially with the $\emptyset$ concept and label respectively). In Figure 3.3, we depict the complete graph corresponding to the semantic graph presented in Figure 3.2. Note that we omit relation labels for clarity purposes.

Formally, this task can be defined as follows. Let $\boldsymbol{x} \in \{0,1\}^{|V| \times |E|}$ be a binary vector indexed by vertices and concepts such that $x_{v,e} = 1$ means that vertex $v \in V$ is tagged with concept $e \in E$. Similarly, let $\boldsymbol{y} \in \{0,1\}^{|A| \times |L|}$ be a binary vector indexed by arcs and labels such that $y_{a,l} = 1$ means that the arc $a \in A$ is labeled with the label $l \in L$. Then, if a tuple $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ satisfies the set of constraints imposed by the semantic representation, it represents a semantic graph anchored on the sentence w, *i.e.* the word w_i corresponding to each vertex $v \in V$ is known. We denote $\mathcal{C}$ the set of all the structures $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ that represent a valid semantic graph. Note that while there can be additional constraints imposed, this means at least having exactly one concept $e \in E$ selected per vertex and one label $l \in L$ selected per arc. Let $\boldsymbol{\mu} \in \mathbb{R}^{|\boldsymbol{x}|}$ and $\boldsymbol{\phi} \in \mathbb{R}^{|\boldsymbol{y}|}$ be two vectors such that $\mu_{v,e}$ is the weight of tagging the vertex $v \in V$ with the concept $e \in E$ and $\phi_{a,l}$ is the weight of labeling the arc $a \in A$ with the label $l \in L$. Given a sentence w, graph-based semantic parsing can be expressed as finding the structure $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}$ of maximum weight:

$$\underset{\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}}{\arg\max}\, \boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y}$$

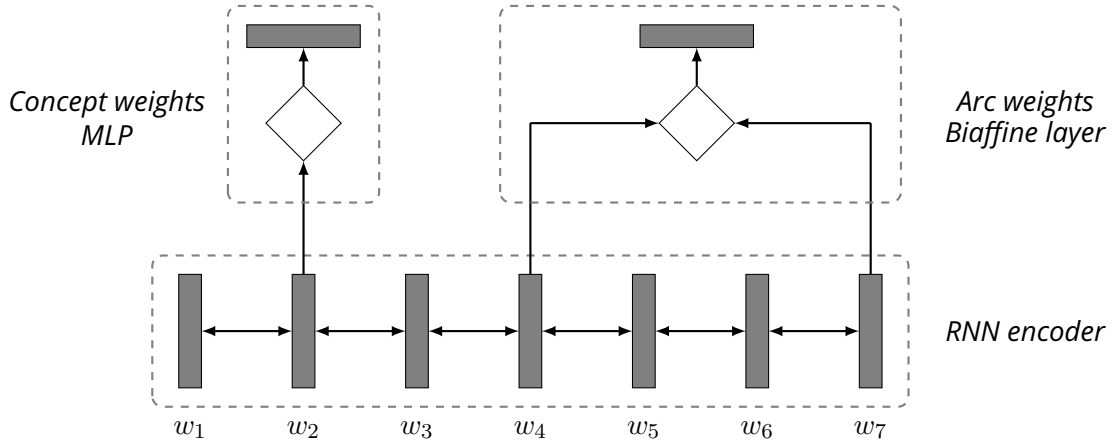

Figure 3.4: Common neural architecture of a graph-based semantic parser. For clarity, we represent the final layers for a single word and a single pair of word respectively. Note that they are applied to each word and each pair of words respectively.

3.3.2 Neural architecture structure

In the formal definition of our task, we introduced two weight vectors $\boldsymbol{\mu}$ and $\boldsymbol{\phi}$ that are produced by a neural architecture. To obtain these vectors, a sentence is first encoded to obtain a contextual representation of each word. Then, these representations are used to compute the vectors $\boldsymbol{\mu}$ and $\boldsymbol{\phi}$. We describe these steps below and illustrate them in Figure 3.4.

Encoder. The sentence is encoded either by a recurrent network, usually a bi-LSTM (Hochreiter and Schmidhuber, 1997), or a self-attentive encoder (Vaswani et al., 2017). This produces a contextual representation $\boldsymbol{h}_i$ for each word w_i in the sentence w.

Concept weights. For each word w_i, its representation $\boldsymbol{h}_i$ is given to a multi-layer perceptron. The output is a vector of dimension $|E|$ that contains a weight $\mu_{v,e}$ for each concept $e \in E$, *i.e.* the weight of tagging the vertex v_i corresponding to w_i with the concept e. Concatenating the output of this network for each word $w_i \in w$ yields $\boldsymbol{\mu}$.

Label weights. For each pair of words w_i and w_j, their representations $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ are given to a biaffine layer (Dozat and Manning, 2017). The output is a vector of dimension $|L|$ that contains a weight $\boldsymbol{\phi}_{v_i \to v_j, l}$ for each label $l \in L$, *i.e.* the weight of adding an arc labeled with l from v_i to v_j where v_i and v_j are the vertices corresponding to w_i and w_j

respectively. Note that in the case where the arcs are not labeled, the output is simply a scalar representing the weight of adding an arc from v_i to v_j.

3.4 Graph-based parsing for compositional generalization

Recent work in semantic parsing has focused on improving compositional generalization. As highlighted in Section 2.4, models relying on the composition of local predictions, like span-based approaches, outperformed sequence-to-sequence architectures. It is thus appealing to explore graph-based parsing as it relies on local predictions, similarly to span-based parsing. In this section, we present the two approaches introduced concurrently to the contributions presented in this thesis: LAGr (Jambor and Bahdanau, 2022) and the application of an existing AM parser (Groschwitz et al., 2018) for compositional generalization.

LAGr parser. LAGr follows the standard pipeline for graph-based parsing by predicting the concepts first and the labels in a second step. The specificity of this parser lies in the fact that each word can produce up to two vertices. It is thus possible to produce graphs with more vertices than words in the sentence. To do so, two separate vectors of weights are computed by the neural network, one for each potential vertex. To predict the arcs, there is a score for each pair of vertices and each label. Thus, it is possible to have an arc between the two vertices that correspond to the same word w_i.

Inference in LAGr is efficient as the prediction for each vertex and arc in the graph is made independently. As this approach outperforms sequence-to-sequence approaches on compositional generalization tasks, it hints that more constrained inference algorithms could lead to even better performances, which we demonstrate with our contributions in Chapters 5 and 6.

AM parser. The AM (Apply-Modify) parser was originally designed for AMR parsing. It differs from other parsers by relying on a linguistically motivated graph composition algebra. In this approach, a supertag is predicted for each word instead of a concept. This supertag represents a graph fragment composed of a labeled vertex as well as its adja-

cent arcs. Between pairs of words, the model predicts whether their respective graph fragments should be composed and which operation (`Apply` or `Modify`) should be used.

For example, in the COGS dataset, the verb "see" produces a vertex tagged with the concept `see`. This vertex has two outgoing arcs labeled `agent` and `theme` respectively. The AM parser will predict a supertag that represents both the vertex and the two arcs. Then, it will predict two `Apply` operations towards the supertags that represent the vertices that are the destination of the `agent` and `theme` arcs.

During inference, the supertags and operations predicted must yield a valid graph. The inference algorithm is thus more constrained than LAGr. As this approach significantly outperforms both sequence-to-sequence architectures and LAGr on the COGS dataset, it confirms that more constrained inference algorithms can be beneficial for compositional generalization.

3.5 Datasets

There is a large variety of existing datasets for semantic parsing. In this section, we introduce the five datasets that are used in the following chapters: GeoQuery, Scan, Clevr, COGS and CFQ. Notably, we explain how their semantic representations can be converted into graphs and how the data splits are constructed for each of them.

3.5.1 GeoQuery

GeoQuery (Zelle and Mooney, 1996) is a dataset containing 880 questions regarding the geography of the United States. The associated semantic representation is a first-order logical form augmented with quantification operations over sets that uses the Prolog syntax. As the approach proposed by Kate et al. (2005) did not handle variables, they also introduced FunQL, a variable-free functional representation for GeoQuery that can be mapped deterministically to the original logical forms. In this representation, each concept is a function that takes other concepts as arguments. We illustrate the two representations in Table 3.1.

Sentence	Name all the rivers in Colorado .
Original	`answer(A,(river(A),loc(A,B),const(B,stateid(colorado))))`
FunQL	`answer(river(loc_2(stateid(colorado))))`
Sentence	How big is Texas ?
Original	`answer(A,(size(B,A),const(B,stateid(texas))))`
FunQL	`answer(size(stateid(texas)))`

Table 3.1: Examples of sentences in the GeoQuery dataset with their original and FunQL representations.

Conversion to graph. To represent a logical form as a graph, we use the FunQL representation as our starting point. We can transform it to a graph representation as follows:

1. A predicate with the argument `all` is a special construction used to designate the set of all the objects of a given type. We first merge the predicate and its argument as a single entity, *e.g.* `city(all)` is converted to the entity `city_all`.

2. Each object is represented by a concept and its name. For example, the state of Texas is represented as `stateid(texas)`. For each of these objects, we create a vertex labeled with the concept, *i.e.* `stateid` in the example, as the name can be recovered by looking at the word the vertex will be anchored on.

3. For every other concept (including the entities created in the first step) e we create a vertex labeled with that concept.

4. Finally, for each argument e' of a concept e, we create an arc from the vertex representing e to the vertex representing e'.

We illustrate this transformation in Figure 3.5. Note that the arcs are not labeled in GeoQuery.

Data splits. The dataset contains 880 queries and the original data split consisted in sampling 540 sentences for the train set, 60 for the dev set and use the remaining 280 queries as the test set. We refer to this split as the IID split. To specifically assess compositional generalization, two additional splits were introduced more recently. The first one, known as the Template split, was created in such a way that two sentences whose programs are identical if entities are anonymized must be in the same set (Finegan-Dollak

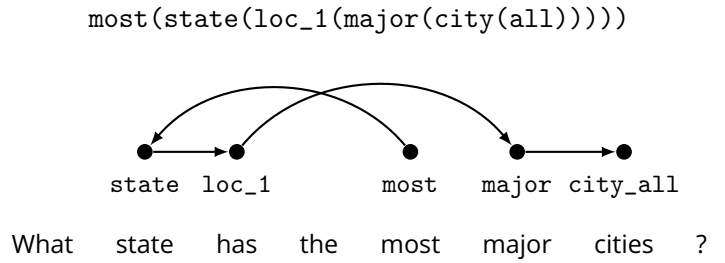

Figure 3.5: **(top)** Semantic analysis of the sentence "What state has the most major cities ?" in the FunQL formalism for GeoQuery. **(bottom)** Graph-based representation of the semantic structure.

et al., 2018). For example, the two FunQL representations `city(loc_2(stateid(texas)))` and `city(loc_2(stateid(alaska)))` are identical if the identifier of the state is removed. This split contains 544 sentences in the train set, 60 in the dev set and 276 in the test set. The second split, known as the Length split, was created by sorting the sentences according to the number of concepts in their FunQL representation (Herzig and Berant, 2021). The 540 sentences with the smallest number of concepts constitute the training set, the dev set contains the following 60 and finally, the test set contains the 280 sentences with the highest number of concepts.

The type and expected arguments of each concept in GeoQuery are detailed in Appendix A.

3.5.2 SCAN

SCAN (Lake and Baroni, 2018) is a dataset originally designed to evaluate compositional generalization in seq2seq models. In this dataset, the goal is to translate a natural language command into a sequence of actions. In a natural language command, an action can be modified with an orientation ("left" or "right"), a number of repetitions ("twice" or "thrice") or by requiring the agent to perform a half or a full turn on itself ("opposite" or "around"). With these modifiers, the number of actions in a sequence can be significantly bigger than the number of words in the natural language command. For example, the command "Run around left thrice" alone will generate 24 actions. Herzig and Berant (2021) proposed an alternative version of this dataset, denoted SCAN-SP, which uses a functional representation. Each word in a sentence is always mapped to a concept with the same name (*e.g.* the word "jump" is mapped to the concept "`jump`"), thus this dataset

Sentence	Original representation	SCAN-SP representation
Turn left after jump twice	`I_JUMP I_JUMP I_TURN_LEFT`	`after(turn(left),twice(jump))`
Look and run left	`I_LOOK I_TURN_LEFT I_RUN`	`and(look,run(left))`
Walk twice after look	`I_LOOK I_WALK I_WALK`	`after(twice(walk),look)`

Table 3.2: Examples of sentences in the SCAN dataset with their original and SCAN-SP representations.

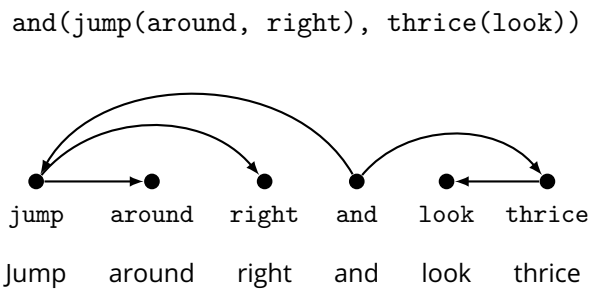

Figure 3.6: **(top)** Semantic analysis of the sentence "Jump around right and look thrice" in the SCAN-SP formalism. **(bottom)** Graph-based representation of the semantic structure.

evaluates whether a model can learn simple composition rules. We illustrate the two representations in Table 3.2.

Conversion to graph. Similarly to the FunQL representation for GeoQuery, the semantic programs in SCAN-SP take the form of a functional language. Thus, we can follow the same procedure to convert the programs into graphs. Note that the first two steps handled the specificities of GeoQuery and are not needed for SCAN. We illustrate the transformation in Figure 3.6.

Data splits. The sentences in this dataset are generated using a grammar without recursion, which means that exactly 20910 unique sentences can be generated. The original split, known as the *simple split*, introduced by Lake and Baroni (2018) sampled 80% of these for the training set and 20% for the test set. We refer to it as the IID split. To evaluate compositional generalization, additional splits were introduced by Loula et al. (2018). In the following chapters of this thesis, we will use two of them: the Right and AroundRight splits. The first one separates the sentences between the training and the test set in such a way that a sentence is in the test set if and only if it contains an action immediately followed by "right" (e.g. "jump right"). Similarly, the test set in the AroundRight split contains

a sentence if and only if it contains an action followed by "around right" (e.g. "turn left and jump around right"). As the splits in this dataset do not contain a dev set, 25% of the training set is usually sampled to be used as a dev set, making it the same size as the test set.

The type and expected arguments of each concept in SCAN are detailed in Appendix B.

3.5.3 Clevr

Clevr (Johnson et al., 2017) contains synthetic questions regarding the properties of objects and the relations between them in synthetic images. Mao et al. (2019) proposed a domain specific language (DSL) for Clevr that takes the form of a functional language, similarly to the two datasets introduced before. In addition to the original dataset, a new test set known as Closure (Bahdanau et al., 2019) was proposed to test compositional generalization. It contains questions generated from new templates.

This representation contains two special concepts: "`filter`" and "`scene`". The first one returns a subset of objects given a set and multiple properties that must be satisfied and the second one returns the set of all the objects present in the scene. They can be problematic to handle as they do not correspond to specific words in the sentence. Herzig and Berant (2021) added them to the representation predicted by their parser in a deterministic fashion.

Conversion to graph. Similarly to GeoQuery, the conversion of Clevr semantic representations requires some pre-processing. The steps to convert them are as follows:

1. The entities "`left`", "`right`", "`front`" and "`behind`" only appear as the first argument of the predicate "`relate`". Thus, when the latter is encountered, we create a vertex labeled with the concatenation of "`relate`" and its first argument, *e.g.* "`relate_left`".

2. The predicate "`filter`" takes multiple properties and a set of objects as arguments. We do not create a vertex for "`filter`" but we create one for each of its arguments. If its final argument is "`scene`", we do not create a vertex for it. Then, we create an arc from the vertex representing each argument to the vertex representing the

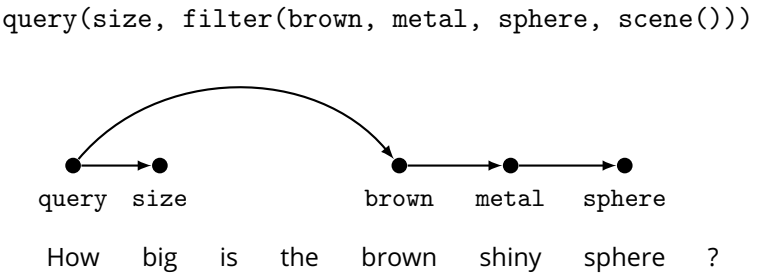

Figure 3.7: **(top)** Semantic analysis of the sentence "How big is the brown shiny sphere" in the Clevr dataset. **(bottom)** Graph-based representation of the semantic structure.

following argument. In step 4, the vertex that will be considered as corresponding to "`filter`" is the vertex of its first argument.

3. For every other concept e, we create a vertex labeled e.

4. Except for the concepts mentionned in the first two steps, for every argument e' of a concept e, we create an arc from the vertex representing e to the vertex representing e'.

We illustrate this transformation in Figure 3.7.

Data splits. The original dataset contains 699689 questions in the train set and 149991 in the development set. The test set cannot be used for semantic parsing as the description of the scenes are not publicly available. In our contributions, we sample 5000 questions from the train set as a development set and use the original development set as a test set, following the approach proposed by Herzig and Berant (2021). The Closure split (Bahdanau et al., 2019) introduces a new test set designed to evaluate systematic generalization. It was constructed based on 7 new query templates and contains 25200 questions, 3600 for each of these templates.

The type and expected arguments of each concept in Clevr are detailed in Appendix C.

3.5.4 COGS

The *Compositional Generalization Challenge based on Semantic Interpretation* (COGS, Kim and Linzen, 2020) dataset has been designed explicitly to evaluate compositional generalization. A semantic structure in COGS is represented as a logical form in which the

*cat(x_1)
AND like.agent(x_2, x_1)
AND like.ccomp(x_2, x_5)
AND prefer.agent(x_5, Emma)
AND prefer.xcomp(x_5, x_7)
AND walk.agent(x_7, Emma)

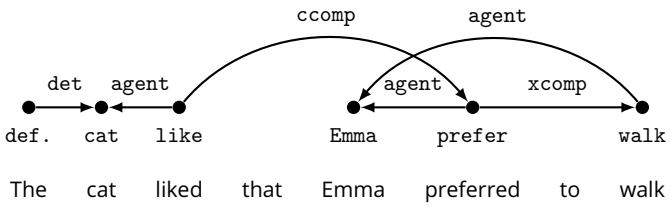

Figure 3.8: **(left)** Semantic analysis of the sentence "The cat liked that Emma preferred to walk" in the COGS formalism. A $*$ denotes definiteness of the following predicate. **(right)** Graph-based representation of the semantic structure. Note that we mark definiteness using an extra vertex anchored on the determiner.

common nouns are identified using unary predicates, proper nouns using constants and verbs as well as relationships between nouns are identified using binary predicates. This is illustrated in Figure 3.8 on the left.

COGS evaluates two distinct types of compositional generalizations: *lexical* and *structural* generalizations. Lexical generalization consists in evaluating a model on known grammatical structures where words are used in a role that was not observed during training, *e.g.* observing a word as the object of a verb while it was only seen as a subject during training. Structural generalization consists in evaluating a model on novel syntactic structures. COGS presents 18 lexical generalization cases and 3 structural generalization cases. Illustrations of these generalizations can be found in Table 3.3.

Conversion to graph. We transform the logical forms found in COGS into graphs with the following steps:

1. For each concept, we create a labeled vertex.

2. For each argument e of a concept e', we create a labeled arc from the vertex representing e' to the vertex representing e.

3. COGS explicitly identifies definiteness of nouns. Therefore, for each definite noun that triggers a concept, we create a vertex v with label `definite` and we create an arc with label `det` from v to the vertex representing the noun's concept. [1] Indefiniteness is marked by the absence of such structure.

[1] Using the determiner as the head of a relation may be surprising for readers familiar with syntactic dependency parsing datasets, but there is no consensus among linguists about the appropriate dependency direction, see *e.g.*, Müller (2016, Section 1.5) for a discussion.

	Training example	Generalization example
Lexical generalizations : novel combinations		
Subject → Object (common)	A **hedgehog** ate the cake.	The baby liked the **hedgehog**.
Subject → Object (proper)	**Lina** gave the cake to Olivia.	A hero shortened **Lina**.
Object → Subject (common)	Henry liked a **cockroach**.	The **cockroach** ate the bat.
Object → Subject (proper)	The creature grew **Charlie**.	**Charlie** worshipped the cake.
Primitive → Subject (common)	**shark**	A **shark** examined the child.
Primitive → Subject (proper)	**Paula**	**Paula** sketched William.
Primitive → Object (common)	Henry liked a **cockroach**.	The **cockroach** ate the bat.
Primitive → Object (proper)	The creature grew **Charlie**.	**Charlie** worshipped the cake.
Primitive → Infinitive	**crawl**	A baby planned to **crawl**.
Lexical generalizations : argument structure and verb class		
Active → Passive	The crocodile **blessed** William.	A muffin **was blessed**.
Passive → Active	The booked **was squeezed**.	The girl **squeezed** the strawberry.
OO transitive → Transitive	Emily **baked**.	The giraffe **baked a cake**.
Unaccusative → Transitive	The glass **shattered**.	Liam **shattered** the jigsaw.
DO dative → PP dative	The girl **teleported** Liam the cookie.	Benjamin **teleported** the cake **to** Isabella.
PP dative → DO dative	Jane **shipped** the cake **to** John.	Jane **shipped** John the cake.
Agent NP → Unaccusative subj.	The cobra **helped** a dog.	The cobra **froze**.
Theme NP → OO transitive subj.	The hippo **decomposed**.	The hippo **painted**.
Theme NP → Unergative subj.	The hippo **decomposed**.	The hippo **giggled**.
Structural generalizations		
Object PP → Subject PP	Noah ate **the cake on the plate**.	**The cake on the table** burned.
Deeper PP recursion	Ava saw the ball **in the bottle**.	Ava saw the ball **in the bottle on the table on the floor**.
Deeper CP recursion	Emma said **that** the cat danced.	Emma said **that** Noah knew **that** Lucas saw **that** the cat danced.

Table 3.3: A list of all the lexical and structural generalization cases from COGS. “OO“ and “DO“ stand for “Object-omitted“ and “Double-object“ respectively.

This transformation is illustrated in Figure 3.8.

Data splits. COGS possesses two data splits, the IID split and the Generalization split. The IID split contains 24155 sentences in the training set, 3000 in the development set and 3000 in the test set. The same training and development sets are used for the Generalization split. The associated test set was constructed by generating exactly 1000 sentences per generalization case, yielding a total of 21000 sentences. Note that for each lexical generalization case, every sentence in the test set employs the same word, *e.g.* the 1000 sentences in the “Primitive → Infinitive” generalization introduce the verb “crawl”. A word used in a lexical generalization can only appear in a single sentence in the training set.

3.5.5 CFQ

The *Compositional Freebase Questions* (CFQ, Keysers et al., 2020) dataset consists of natural language questions associated to corresponding SPARQL queries that can be executed against the Freebase knowledge base (Bollacker et al., 2008). A SPARQL query is composed of a `SELECT` and a `WHERE` clause. The `SELECT` clause can either be `SELECT COUNT(*)` for yes/no questions or `SELECT DISTINCT ?x0` for wh- questions. The `WHERE` clause contains predicates with one (*e.g.* `?x0 actor`) or two entities (*e.g.* `M1 starring ?x0`) as arguments or filter constraints requiring two entities to be distinct (*e.g.* `FILTER (?x0 != ?x1)`). We illustrate this in Figure 3.9 on the left.

Conversion to graphs. We transform the SPARQL queries into graphs with the following steps:

1. In the case of wh- questions, we replace any mention of `?x0` in the `WHERE` clause by `select_?x0` which allows us to discard the `SELECT` clause.

2. The `nationality` and `gender` concepts take an entity as their left argument and the corresponding nationality or gender as their right argument. To reduce the amount of vertices, for each possible right argument, we create a distinct copy of these concepts that only take a left argument (*e.g.* `?x0 nationality swedish` becomes `?x0 nationality_swedish`).

3. Similarly to previous work (Furrer et al., 2020; Guo et al., 2020; Jambor and Bahdanau, 2022), we merge concepts that share an identical argument (*e.g.* `M1 starring ?x0. M2 starring ?x0` becomes `[M1, M2] starring ?x0`).

4. For each distinct concept in the `WHERE` clause, we create a vertex v in the semantic graph. For variables (`?x`), we assign a generic concept `var` to v. In the case where entities were merged together, we create a vertex tagged with a `conjunction` concept and an arc labeled `conjunction` towards each vertex corresponding to an entity.

5. For each concept e with arguments, we create arcs with the `agent` and `theme` labels towards the vertices representing its left and right arguments respectively. In the

```
SELECT COUNT(*) {
M1 ns:produce ?x0.
M1 ns:edit ?x0.
M2 ns:produce ?x0.
M2 ns:edit ?x0.
?x0 ns:film }
```

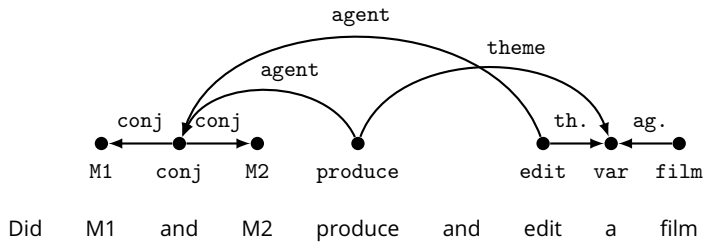

Figure 3.9: **(left)** Semantic analysis of the sentence "Did M1 and M2 produce and edit a film" in the CFQ formalism. **(right)** Graph-based representation of the semantic structure. Note that we abbreviate `agent` and `theme` as `ag.` and `th.` respectively for readability purposes.

case where an arguments consists of merged entities, the arc is created towards the vertex tagged with `conjunction` instead.

We illustrate this transformation in Figure 3.9.

Data splits. CFQ contains 4 data splits. The first one is the IID split which contains 95744 sentences in the training set and 11967 in the dev and test sets. The three other splits, designed to evaluate compositional generalization, are known as *Maximum Compound Divergence* (MCD) 1, 2 and 3. These splits were constructed in such a way that they satisify two properties:

1. The divergence between the frequency of apparition of each word in the training set on one hand and the dev and test sets on the other hand is minimized.
2. The divergence between the frequency of apparition of combinations of elements in the training set on one hand and the dev and test sets on the other hand is maximized.

Thus, these splits require a model to generalize to novel combinations of elements as well as new syntactic structures at test time. Each of them contains 95743 sentences in the training set while the dev and test sets contain 11968 sentences each.

3.6 Conclusion

In this chapter, we introduced graph-based approaches for semantic parsing. Notably, we presented the motivation for relying on these approaches and formulated the task as an

optimization problem. We highlighted that the inference algorithm is heavily dependent on the set of constraints imposed on the semantic representation. Thus, the approaches presented in the following chapters will be motivated by the constraints that are imposed on our semantic representations.

We also introduced the five semantic parsing datasets that we used in our experiments. While their respective semantic formalisms are logical forms and database queries, we proposed a deterministic transformation into graphs for each of them. This highlights how graphs allow for a unified approach for semantic parsing with limited manual work being required.

Chapter 4

Combinatorial optimization and training

In Chapter 3, we introduced graph-based semantic parsing. As the output of these models is a structure, computing the training objective can be tricky. A majority of recent works rely on approximations to train their neural architectures. For example, both the span-based parser of Herzig and Berant (2021) and the graph-based parser of Jambor and Bahdanau (2022) are trained by computing an approximation of the structure of maximum weight and maximizing the likelihood of this approximation. In both cases, there is no guarantee that the chosen structure is the one of maximum weight.

Our novel contribution is a mathematically motivated training objective for graph-based semantic parsing. In Section 4.1, we define the objective and discuss the issue of marginalizing over a set of structures. To bypass this issue, we propose a surrogate objective function that is a lower bound to the original objective. However, we prove that computing one of the terms in this surrogate objective is NP-hard and can be cast as an optimization problem. In Section 4.2, we introduce an optimization framework. It relies on the conditional gradient method, also known as the Frank-Wolfe algorithm (Frank and Wolfe, 1956; Levitin and Polyak, 1966; Lacoste-Julien and Jaggi, 2015), an iterative algorithm that solves optimization problems. This method requires an efficient subroutine over the search space of the problem, which is not the case for our problem. In the rest of this section, we propose an approach based on Nesterov (2005) and Yurtsever et al. (2018)

to handle problematic constraints and define a new search space for which a subroutine exists. Finally, in Section 4.3, we propose two ways to tackle our NP-hard problem. The first relies on the optimization framework introduced in the previous section for which we present an efficient algorithm. The second approach casts our problem as MAP inference in a factor graph. Parts of the contributions in this chapter were published in Petit and Corro (2023) and Petit et al. (2023).

4.1 Training objective

Remember that in Section 3.3, we represented a graph anchored on a sentence as a structure $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ where $\boldsymbol{x}$ and $\boldsymbol{y}$ represent the concepts and labels assigned to the vertices and arcs respectively. Let $\mathcal{C}$ be the set of all the valid semantic structures $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. Given the weight vectors $\boldsymbol{\mu} \in \mathbb{R}^{|\boldsymbol{x}|}$ and $\boldsymbol{\phi} \in \mathbb{R}^{|\boldsymbol{y}|}$, we can define the likelihood of a structure $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}$ via the Boltzmann distribution:

$$p_{\boldsymbol{\mu},\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}) = \exp(\boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y} - c(\boldsymbol{\mu}, \boldsymbol{\phi})),$$

where $c(\boldsymbol{\mu}, \boldsymbol{\phi})$ is the log-partition function:

$$c(\boldsymbol{\mu}, \boldsymbol{\phi}) = \log \sum_{\langle \boldsymbol{x}', \boldsymbol{y}' \rangle \in \mathcal{C}} \exp(\boldsymbol{\mu}^\top \boldsymbol{x}' + \boldsymbol{\phi}^\top \boldsymbol{y}').$$

During training, we aim to maximize the log-likelihood of the training dataset. The log-likelihood of an observation $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ is defined as:

$$\begin{aligned} \ell(\boldsymbol{\mu}, \boldsymbol{\phi}; \boldsymbol{x}, \boldsymbol{y}) &= \log p_{\boldsymbol{\mu},\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}) \\ &= \boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y} - c(\boldsymbol{\mu}, \boldsymbol{\phi}). \end{aligned}$$

However, in practice we usually do not have access to a structure $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. A common occurence is to have instead the semantic graph without concept anchoring. The difference is illustrated in Figure 4.1. We consider our training signal to be the set of all the structures $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ that induce the gold semantic graph, which we denote $\mathcal{C}^*$. We call this setting weakly-supervised and the training objective is the marginalization over all the structures

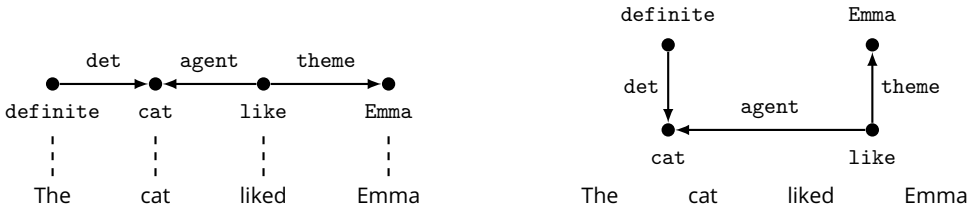

Figure 4.1: **(left)** A semantic graph with its vertices anchored on the words in the sentence. The anchorings are represented by dashed lines. **(right)** The same semantic graph without anchoring on the sentence.

in $\mathcal{C}^*$:

$$\begin{aligned}
\widetilde{\ell}(\boldsymbol{\mu}, \boldsymbol{\phi}; \mathcal{C}^*) &= \log \sum_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^*} p_{\boldsymbol{\mu}, \boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}) \\
&= \log \sum_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^*} \exp\Big(\boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y} - c(\boldsymbol{\mu}, \boldsymbol{\phi})\Big) \\
&= \left(\log \sum_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^*} \exp(\boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y})\right) - c(\boldsymbol{\mu}, \boldsymbol{\phi})
\end{aligned}$$

Unfortunately, computing both terms of this objective is intractable as it requires summing over $\mathcal{C}^*$ and $\mathcal{C}$ respectively. Instead, we rely on a lower bound as a surrogate objective function. This approach is common in machine learning (Bishop, 2006). To this end, we aim to derive a lower bound to the first term and an upper bound (because it is negated in $\widetilde{\ell}$) to the second term.

4.1.1 Variational formulation of a LogSumExp function

Note that each term in the weakly-supervised loss is a LogSumExp (LSE) function. To derive bounds of this function, we first express it under its variational formulation. To do so, we will rely on its Fenchel conjugate defined below.

> **Definition 4.1: Convex set and convex function**
>
> A set $X \subseteq \mathbb{R}^k$ is convex if and only if
>
> $$\forall \boldsymbol{x}, \boldsymbol{x}' \in X, \epsilon \in [0, 1] : \epsilon \boldsymbol{x} + (1 - \epsilon)\boldsymbol{x}' \in X$$

A function $f : X \to \mathbb{R}$ is convex if and only if X is a convex set and

$$\forall \boldsymbol{x}, \boldsymbol{x}' \in X, \epsilon \in [0,1] : f(\epsilon \boldsymbol{x} + (1-\epsilon)\boldsymbol{x}') \leq \epsilon f(\boldsymbol{x}) + (1-\epsilon) f(\boldsymbol{x}')$$

Definition 4.2: Proper function

A convex function $f : X \to \mathbb{R}$ is proper if and only if

$$\forall \boldsymbol{x} \in X : f(\boldsymbol{x}) > -\infty \qquad \text{and} \qquad \exists \boldsymbol{x} \in X : f(\boldsymbol{x}) < +\infty$$

Definition 4.3: Closed function

A function $f : X \to \mathbb{R}$ is closed if and only if for each $\alpha \in \mathbb{R}$, the set $\{\boldsymbol{x} \in X | f(\boldsymbol{x}) \leq \alpha\}$ is a closed set.

Definition 4.4: Fenchel conjugate

Let $f : \mathbb{R}^k \to \mathbb{R} \cup \{\infty\}$ be a function. The Fenchel conjugate of f is the convex function $f^* : \mathbb{R}^k \to \mathbb{R} \cup \{\infty\}$ defined as follows:

$$f^*(\boldsymbol{t}) = \sup_{\boldsymbol{u} \in \operatorname{dom} f} \boldsymbol{u}^\top \boldsymbol{t} - f(\boldsymbol{u})$$

Theorem 4.5: Fenchel-Moreau theorem

Let $f : \mathbb{R}^k \to \mathbb{R} \cup \{\infty\}$ be a function. The biconjugate of f is the function $f^{**} : \mathbb{R}^k \to \mathbb{R} \cup \{\infty\}$ defined as follows:

$$f^{**}(\boldsymbol{u}) = \sup_{\boldsymbol{t} \in \operatorname{dom} f^*} \boldsymbol{t}^\top \boldsymbol{u} - f^*(\boldsymbol{t})$$

If f is convex, closed and proper, then $f^{**} = f$.

Proof. A proof of Theorem 4.5 can be found in Beck (2017, Theorem 4.8). □

Definition 4.6: Simplex

The simplex Δ^k of dimension $k-1$ is the set of all probability vectors of dimension k. Formally, it can be written as

$$\Delta^k = \{\boldsymbol{p} \in \mathbb{R}^k \mid \sum_i p_i = 1 \quad \text{and} \quad \forall i : p_i \geq 0\}$$

As a LSE function is convex, closed and proper (Beck, 2017, Sections 4.4.10 and 4.4.11), we can express it via its Fenchel biconjugate. Let $f : \mathbb{R}^k \to \mathbb{R}$ be a LSE function, *i.e.* $f(\boldsymbol{u}) = \log \sum_i \exp(u_i)$. Its Fenchel conjugate is:

$$f^*(\boldsymbol{t}) = \max_{\boldsymbol{u} \in \mathbb{R}^k} \boldsymbol{u}^\top \boldsymbol{t} - f(\boldsymbol{u})$$

By the first order optimality conditions, we can compute the optimal value by setting the derivative with respect to each element u_i to 0:

$$\begin{aligned}
0 &= \frac{\partial}{\partial u_i} \boldsymbol{u}^\top \boldsymbol{t} - f(\boldsymbol{u}) \\
0 &= \frac{\partial}{\partial u_i} \boldsymbol{u}^\top \boldsymbol{t} - \log \sum_i \exp(u_i) \\
0 &= t_i - \frac{\exp(u_i)}{\sum_j \exp(u_j)} \\
t_i &= \frac{\exp(u_i)}{\sum_j \exp(u_j)}
\end{aligned}$$

Note that $\boldsymbol{t}$ is the softmax of $\boldsymbol{u}$. Thus, it sums to 1 and every element t_i is positive. This means that the domain of f^* is Δ^k.

$$\begin{aligned}
t_i &= \frac{\exp(u_i)}{\exp(f(\boldsymbol{u}))} \\
\exp(u_i) &= t_i \exp(f(\boldsymbol{u})) \\
u_i &= \log(t_i \exp(f(\boldsymbol{u}))) \\
u_i &= \log(t_i) + f(\boldsymbol{u})
\end{aligned}$$

We compute the Fenchel conjugate by rewriting $\boldsymbol{u}$:

$$\begin{aligned} f^*(\boldsymbol{t}) &= \sum_i (t_i \times (\log t_i + f(\boldsymbol{u}))) - f(\boldsymbol{u}) \\ &= \sum_i t_i \log t_i + \sum_i t_i f(\boldsymbol{u}) - f(\boldsymbol{u}) \\ &= \sum_i t_i \log t_i + f(\boldsymbol{u}) - f(\boldsymbol{u}) \\ &= \sum_i t_i \log t_i \end{aligned}$$

This function is the negative Shannon entropy. We will denote the Shannon entropy as $H[\boldsymbol{t}] = -\sum_i t_i \log t_i$. Using these results, we can rewrite a LSE function via its Fenchel biconjugate as follows:

$$\begin{aligned} f(\boldsymbol{u}) &= f^{**}(\boldsymbol{u}) \\ &= \max_{\boldsymbol{t} \in \Delta^k} \boldsymbol{t}^\top \boldsymbol{u} - f^*(\boldsymbol{t}) \\ &= \max_{\boldsymbol{t} \in \Delta^k} \boldsymbol{t}^\top \boldsymbol{u} - \sum_i t_i \log t_i \\ &= \max_{\boldsymbol{t} \in \Delta^k} \boldsymbol{t}^\top \boldsymbol{u} + H[\boldsymbol{t}] \end{aligned}$$

These computations are inspired by Boyd and Vandenberghe (2004, Example 3.25), Wainwright and Jordan (2008, Section 3.6) and Beck (2017, Sections 4.4.10 and 4.4.11).

> **Definition 4.7: Variational formulation of the LSE over a set of structures**
>
> Let $\mathcal{D}$ be a set of structures $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. Let $\boldsymbol{U}$ be a matrix such that each row contains a pair $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{D}$ and $\Delta^{|\mathcal{D}|}$ be the simplex of dimension $|\mathcal{D}| - 1$. Note that $\left(\boldsymbol{U} \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\phi} \end{bmatrix} \right)$ is a vector in which each element is the weight of a structure $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{D}$. The LSE over the set of structures $\mathcal{D}$ can be equivalently rewritten as:
>
> $$\log \sum_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in D} \exp(\boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y}) = \max_{\boldsymbol{p} \in \Delta^{|\mathcal{D}|}} \boldsymbol{p}^\top \left(\boldsymbol{U} \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\phi} \end{bmatrix} \right) + H[\boldsymbol{p}]$$

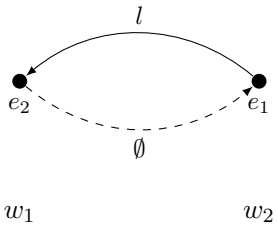

Figure 4.2: Semantic structure represented by the row in red in Example 4.8. Each tagged vertex is anchored on the word below.

Example 4.8: Construction of a matrix of structures

Let $w = (w_1, w_2)$ be a sentence, $E = \{e_1, e_2, \emptyset\}$ the set of concepts and $L = \{l, \emptyset\}$ the set of labels. Let $G = \langle V, A \rangle$ be the complete graph constructed over w where $V = \{1, 2\}$ and $A = \{1 \to 2, 2 \to 1\}$. A matrix $\boldsymbol{U}$ such that each row represents a valid semantic graph over w can be constructed as follows:

$$\boldsymbol{U} = \begin{array}{c} \begin{matrix} x_{1,e_1} & x_{1,e_2} & x_{1,\emptyset} & x_{2,e_1} & x_{2,e_2} & x_{2,\emptyset} & y_{1\to2,l} & y_{1\to2,\emptyset} & y_{2\to1,l} & y_{2\to1,\emptyset} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ \vdots & & & \vdots & & & \vdots & & & \vdots \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ \vdots & & & \vdots & & & \vdots & & & \vdots \end{bmatrix} \end{array}$$

The row in red corresponds to the graph depicted in Figure 4.2.

In the following subsections, we derive an upper bound and a lower bound using the variational formulation introduced in Definition 4.7.

4.1.2 LogSumExp upper bound

We first derive an upper bound to the log-partition function. Based on the formulation in Definition 4.7, $\boldsymbol{p}$ is a probability vector over the set of structures $\mathcal{C}$. As it has an exponential size, this formulation remains impractical.

Definition 4.9: Marginal polytope

Let $\mathcal{D}$ be a set of structures. Let $\boldsymbol{U}$ be a matrix such that each row contains a structure in $\mathcal{D}$ and $\Delta^{|\mathcal{D}|}$ be the simplex of dimension $|\mathcal{D}| - 1$. The marginal polytope $\mathcal{M} = \text{conv}(\mathcal{D})$ is the convex hull of the structures in $\mathcal{D}$. It can be expressed as follows:

$$\mathcal{M} = \{\boldsymbol{p}^\top \boldsymbol{U} \quad | \quad \boldsymbol{p} \in \Delta^{|\mathcal{D}|}\}$$

Using Definition 4.9, we can rewrite the variational formulation over $\mathcal{C}$ as:

$$\log \sum_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in C} \exp(\boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y}) = \max_{\boldsymbol{m} \in \mathcal{M}} \boldsymbol{m}^\top \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\phi} \end{bmatrix} + H_{\mathcal{M}}[\boldsymbol{m}]$$

where $\boldsymbol{m}$ is equivalent to $\boldsymbol{p}^\top \boldsymbol{U}$ and $H_{\mathcal{M}}$ is a joint entropy function defined such that the equality holds. In this reformulation, the maximization acts on the marginal probabilities of vertices and arcs in the graph, therefore it has a polynomial number of variables. Additional details can be found in Wainwright and Jordan (2008, Section 5.2.1) and Blondel et al. (2020, Section 7).

Unfortunately, this optimization problem is hard to solve as $\mathcal{M}$ cannot be caracterized in an explicit manner and $H_{\mathcal{M}}$ is defined indirectly and lacks a polynomial closed form (Wainwright and Jordan, 2008, Section 3.7). However, Cover (1999, Property 4) demonstrated that decomposing the entropy term over each variable independently yielded an upper bound to the joint entropy, *i.e.* $H_{\mathcal{M}}[\boldsymbol{m}] \leq H[\boldsymbol{m}]$. We can derive an upper bound to the log-partition function by decomposing the entropy term and using an outer approximation to the marginal polytope $\mathcal{L} \supseteq \mathcal{M}$, *i.e.* increasing the search space:

$$\log \sum_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in C} \exp(\boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y}) \leq \max_{\boldsymbol{m} \in \mathcal{L}} \boldsymbol{m}^\top \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\phi} \end{bmatrix} + H[\boldsymbol{m}]$$

In particular, we observe that each structure $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}$ has exactly one tag assigned to each vertex and one label assigned to each arc. We denote $\mathcal{C}^{(\text{one})}$ the set of all the structures $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ that satisfy these constraints (note that not all of them correspond to a valid semantic graph). By using $\mathcal{L} = \text{conv}(\mathcal{C}^{(\text{one})})$ as an outer approximation to the marginal

polytope, the optimization problem can be viewed as a sum of independent maximization problems over each vertex and arc. As each of these problems is the variational formulation of a LSE function, the upper bound on $c(\boldsymbol{\mu}, \boldsymbol{\phi})$ can be expressed as a sum of LSE functions, which is tractable:

$$\begin{aligned} c(\boldsymbol{\mu}, \boldsymbol{\phi}) &\leq \max_{\boldsymbol{m} \in \mathcal{L}} \sum_v \left(\sum_e m_{v,e} \mu_{v,e} \right) + \sum_a \left(\sum_l m_{a,l} \phi_{a,l} \right) + H[\boldsymbol{m}] \\ &= \sum_v \left(\log \sum_e \exp(\mu_{v,e} x_{v,e}) \right) + \sum_a \left(\log \sum_l \exp(\phi_{a,l} y_{a,l}) \right) \end{aligned}$$

This approximation can be understood as a generalization of the head selection loss used in dependency parsing (Zhang et al., 2017). Although it may not result in a Bayes consistent loss (Corro, 2023), it works well in practice.

4.1.3 LogSumExp lower bound

We now derive a lower bound to the LSE function over $\mathcal{C}^*$. Intuitively, as the formulation presented in Definition 4.7 takes the form of a maximization problem over a vector $\boldsymbol{p} \in \Delta^{|\mathcal{C}^*|}$, any value of $\boldsymbol{p}$ in the search space will yield a lower bound. We choose to maximize this lower bound using a vector that gives a probability of one to the structure of maximum weight, as in "hard" EM (Neal and Hinton, 1998, Section 6). Note that in this particular case, the Shannon entropy term is equal to 0. Thus, the lower bound is the weight of the chosen structure.

Computing the structure of maximum weight. For a given sentence w, let $G = \langle V, A \rangle$ be a complete graph constructed over w and $G' = \langle V', A' \rangle$ be a semantic graph. We aim to tag the vertices and label the arcs of a subgraph of G such that it is identical to G'. This is equivalent to anchoring each vertex $v' \in V'$ on a vertex of V such that there is at most one vertex anchored on each vertex $v \in V$ and where the weight of an anchoring is defined as follows:

- For each vertex $v' \in V'$ tagged with a concept e and anchored on a vertex $v \in V$, we add the weight $\mu_{v,e}$ of tagging v with e.
- For each arc $u' \rightarrow v' \in A'$ with the label l, let u and v be the vertices in V that u'

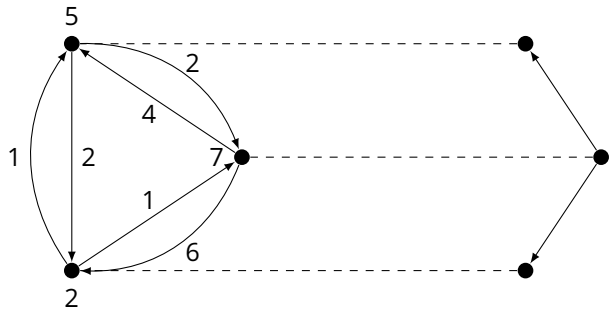

Figure 4.3: A weighted complete graph with 3 vertices is depicted on the left. Dashed lines represent the anchoring of the vertices of the graph on the right over the complete graph. For readability purposes, we assume that all the vertices have the same concept and all arcs have the same label. The weight of this anchoring is 24 (14 for the vertices plus 4 and 6 for the two arcs).

and v' are respectively anchored on. We add the weight $\phi_{u \to v, l}$ of labeling the arc $u \to v \in A$ with l.

Note that this consists in computing a partial alignment between the two graphs. If we did not have arc weights, this could be solved via the Kuhn-Munkres (Kuhn, 1955) or the Jonker-Volgenant (Jonker and Volgenant, 1988; Crouse, 2016) algorithms. However, introducing arc weights makes the use of these algorithms impossible. We illustrate the computation of an anchoring weight in Figure 4.3.

Theorem 4.10: NP-hardness of the anchoring of maximum weight

Computing the anchoring of maximum weight of a semantic graph G' on a graph G is NP-hard.

Proof. We prove Theorem 4.10 by reducing the maximum directed Hamiltonian path problem, which is known to be NP-hard (Garey and Johnson, 1979, Appendix A1.3), to the anchoring of maximum weight.

Let $G = \langle V, A, \boldsymbol{\psi} \rangle$ be a directed weighted graph where $V = \{1, ..., n\}$ and $\boldsymbol{\psi} \in \mathbb{R}^{|A|}$ are arc weights. The maximum Hamiltonian path problem aims to compute the subset of arcs $B \subseteq A$ such that every vertex $v \in V$ is the extremity of at least one arc and the set of arcs B form a path of maximum weight where its weight is defined as $\sum_{a \in B} \psi_a$.

We construct a directed complete graph $G' = \langle V', A' \rangle$ such that there is a vertex $v' \in V'$ corresponding to each vertex $v \in V$. Let e be a concept and l an arc label. The weight of tagging a vertex $v' \in V'$ with e is 0. The weight of labeling an arc $u' \to v' \in A'$ with l is

$\psi_{u \to v}$ if there is an arc $u \to v \in A$ and $-\infty$ otherwise.

We construct a labeled directed graph $G'' = \langle V'', A'' \rangle$ such that $V'' = \{1'', \ldots, n''\}$, $A'' = \{i'' \to (i+1)'' \mid 1 \leq i \leq n\}$, each vertex $v'' \in V''$ is tagged with e and each arc $a'' \in A''$ is labeled with l.

As such, there is a one-to-one correspondence between solutions of the maximum Hamiltonian path problem on graph G and solutions of the anchoring of maximum weight of G'' on G'. Note that if the anchoring found has a weight equal to $-\infty$, there is no Hamiltonian path in G. Thus, the anchoring of maximum weight is NP-hard. □

Therefore, we propose an optimization-based approach to compute the structure of maximum weight. The problem requires each vertex $v \in V$ to be the anchor of at most one vertex $v' \in V'$. We denote $\mathcal{C}^*_{\text{(relaxed)}}$ a set of structures $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ where a vertex $v \in V$ can be the anchor of multiple vertices $v' \in V'$, *i.e.* a word in w can be the anchor of multiple concepts. Computing the structure of maximum weight in $\mathcal{C}^*$ is therefore equivalent to solving the following optimization problem:

$$\begin{aligned} \max_{\boldsymbol{x},\boldsymbol{y}} \quad & \mu^\top \boldsymbol{x} + \phi^\top \boldsymbol{y} \\ \text{s.t.} \quad & \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^*_{\text{(relaxed)}} \\ & \sum_{e \in E} x_{v,e} \leq 1 \qquad \forall v \in V \end{aligned}$$

In the remainder of this chapter, we propose a method that relies on solving the problem on the set $\mathcal{C}^*_{\text{(relaxed)}}$ to find the optimal solution in $\mathcal{C}^*$ and prove that solving the optimization problem on the set $\mathcal{C}^*_{\text{(relaxed)}}$, *i.e.* if the second constraint is ignored, is easy. Thus, we will obtain our training objective.

4.2 Optimization framework with constraint relaxation

In this section, we introduce the framework that we will use to solve the optimization problem presented above.

4.2.1 Conditional gradient method

Let $\mathcal{C} \subseteq \mathbb{R}^k$ be a set, $\mathcal{A} = \text{conv}(\mathcal{C})$ be its convex hull and $f : \mathcal{A} \to \mathbb{R}$ be a smooth, convex and differentiable function. The conditional gradient method, also known as the Frank-Wolfe algorithm (Frank and Wolfe, 1956; Levitin and Polyak, 1966; Lacoste-Julien and Jaggi, 2015), is an algorithm that solves constrained optimization problems of the form:

$$\max_{\boldsymbol{z} \in \mathcal{A}} f(\boldsymbol{z})$$

Unlike other constrained optimization methods, the Frank-Wolfe algorithm does not require to compute projections onto the feasible set $\mathcal{A}$ which is, in most cases, computationally expensive. Given an initial guess $\boldsymbol{z}^{(0)}$, it iteratively converges towards the optimal solution by constructing convex combinations $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \ldots$ of elements in $\mathcal{A}$. To do so, it relies on a linear problem known as the Linear Maximization Oracle (LMO) which is defined as follows:

$$\text{lmo}_{\mathcal{C}}(\boldsymbol{\lambda}) = \arg\max_{\boldsymbol{z} \in \mathcal{C}} \boldsymbol{\lambda}^\top \boldsymbol{z}$$

At each time step k, the update direction is given by computing $\boldsymbol{d} = \text{lmo}_{\mathcal{C}}\left(\nabla f(\boldsymbol{z}^{(k)})\right) - \boldsymbol{z}^{(k)}$. Then, the next convex combination is $\boldsymbol{z}^{(k+1)} = \boldsymbol{z}^{(k)} + \gamma \boldsymbol{d}$ where γ is the stepsize defined as $\arg\max_{\gamma \in [0,1]} f(\boldsymbol{z}^{(k)} + \gamma \boldsymbol{d})$. A pseudo-code of the conditional gradient method is given in Algorithm 1.

Computing the optimal stepsize does not always have a closed form solution. In that case, γ can either follow a pre-determined strategy or be approximated. A common pre-determined strategy is to use $\gamma = \frac{2}{2+k}$. This does not require any additional computation and possesses the same worst-case complexity bounds as adaptive strategies (Dunn and Harshbarger, 1978). As f is convex, γ can also be approximated via the bisection algorithm. This approach requires less steps to converge towards the solution in practice.

Algorithm 1 Frank-Wolfe algorithm

function FW($\mathcal{A}, \epsilon$)
  Let $\boldsymbol{z}^{(0)} \in \mathcal{A}$
  for $k \in \{0....K\}$ **do** ▷ Where K is the maximum number of iterations
    $\boldsymbol{d} \leftarrow \left(\text{lmo}_{\mathcal{C}}(\nabla f(\boldsymbol{z}^{(k)}))\right) - \boldsymbol{z}^{(k)}$ ▷ Compute the update direction
    if $\nabla f(\boldsymbol{z}^{(k)})^\top \boldsymbol{d} \leq \epsilon$ **then return** $\boldsymbol{z}^{(k)}$ ▷ If the dual gap is small, $\boldsymbol{z}^{(k)}$ is (almost) optimal
    $\gamma \in \arg\max_{\gamma \in [0,1]} f(\boldsymbol{z}^{(k)} + \gamma \boldsymbol{d})$ ▷ Compute or approximate the optimal stepsize
    $\boldsymbol{z}^{(k+1)} = \boldsymbol{z}^{(k)} + \gamma \boldsymbol{d}$ ▷ Update the current point
  return $\boldsymbol{z}^{(k)}$

4.2.2 Constraint relaxation and smoothing

The conditional gradient method is used to solve optimization problems over convex sets. However in our case, the optimization problem takes the form:

$$\begin{aligned} \max_{\boldsymbol{z} \in \mathcal{C}} \quad & f(\boldsymbol{z}) \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{z} \leq \boldsymbol{b} \end{aligned}$$

where $\mathcal{C}$ is a set over which there exists an efficient LMO that we will present in Section 4.3. However, the additional constraints $\boldsymbol{A}\boldsymbol{z} \leq \boldsymbol{b}$ prevent the existence of such an oracle. In this setting, an appealing approach is to introduce the problematic constraints as penalties in the objective instead. Let S be the set of all the vectors that satisfy these constraints. We rely on the indicator function of S defined below.

> **Definition 4.11: Indicator function of a set**
>
> Given a set S, its indicator function is defined as
>
> $$\delta_S(\boldsymbol{u}) = \begin{cases} 0 & \text{if } \boldsymbol{u} \in S, \\ +\infty & \text{otherwise} \end{cases}$$

Using the indicator function, the problem can be rewritten as:

$$\max_{\boldsymbol{z} \in \mathcal{C}} f(\boldsymbol{z}) - \delta_S(\boldsymbol{A}\boldsymbol{z})$$

However, the objective is no longer smooth due to the indicator function, preventing the

use of the conditional gradient method. We rely on the framework proposed by Yurtsever et al. (2018) where the indicator function is replaced by a smooth approximation.

> **Theorem 4.12: Fenchel conjugate of an indicator function**
>
> For a given set S, the Fenchel conjugate of its indicator function δ_S is a convex function known as its support function σ_S. This function is defined as
>
> $$\sigma_s(\boldsymbol{u}) = \max_{\boldsymbol{t} \in S} \boldsymbol{u}^\top \boldsymbol{t}$$

Proof. Let S be a set with δ_S and σ_S respectively its indicator and support functions. The Fenchel conjugate of δ_S is

$$\delta_S^*(\boldsymbol{t}) = \max_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{t} - \delta_S(\boldsymbol{u})$$

We distinguish two cases:

- For a given $\boldsymbol{u} \in S$, we have $\boldsymbol{u}^\top \boldsymbol{t} - \delta_S(\boldsymbol{u}) = \boldsymbol{u}^\top \boldsymbol{t}$.
- For a given $\boldsymbol{u} \notin S$, we have $\boldsymbol{u}^\top \boldsymbol{t} - \delta_S(\boldsymbol{u}) = -\infty$.

Thus, we can rewrite the indicator function as

$$\begin{aligned} \delta_S^*(\boldsymbol{t}) &= \max_{\boldsymbol{u} \in S} \boldsymbol{u}^\top \boldsymbol{t} \\ &= \sigma_S(\boldsymbol{t}) \end{aligned}$$

□

Using Theorem 4.12, we rewrite the indicator function via its Fenchel biconjugate:

$$\delta_S(\boldsymbol{A}\boldsymbol{z}) = \delta_S^{**}(\boldsymbol{A}\boldsymbol{z}) = \max_{\boldsymbol{u} \in S} \boldsymbol{u}^\top (\boldsymbol{A}\boldsymbol{z}) - \sigma_S(\boldsymbol{u})$$

It can be smoothed by adding a β-parameterized convex regularizer $-\frac{\beta}{2}\|\cdot\|_2^2$:

$$\delta_{S,\beta}^{**}(\boldsymbol{A}\boldsymbol{z}) = \max_{\boldsymbol{u} \in S} \boldsymbol{u}^\top (\boldsymbol{A}\boldsymbol{z}) - \sigma_S(\boldsymbol{u}) - \frac{\beta}{2}\|\boldsymbol{u}\|_2^2$$

where $\beta > 0$ controls the quality and the smoothness of the approximation (Nesterov, 2005). We can then use the conditional gradient method to solve the following problem instead:

$$\max_{\boldsymbol{z} \in \mathcal{C}} f(\boldsymbol{z}) - \delta^{**}_{S,\beta}(\boldsymbol{A}\boldsymbol{z})$$

In the remainder of this thesis, we follow Yurtsever et al. (2018) and use $\beta^{(k)} = \frac{\beta^{(0)}}{\sqrt{k+1}}$ where k is the iteration number and $\beta^{(0)} = 1$. We detail the computations that yield a closed form to $\delta^{**}_{S,\beta}(\boldsymbol{A}\boldsymbol{z})$ in Appendix D.

Equality case. When the additional constraints are equalities, we have $S = \{\boldsymbol{b}\}$. In that case, we demonstrate in Appendix D that the smoothed indicator function can be expressed as:

$$\delta^{**}_{S,\beta}(\boldsymbol{A}\boldsymbol{z}) = \frac{1}{2\beta}\|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}\|_2^2$$

In this case, we have a quadratic penalty term in the objective. Note that this term is similar to the term introduced in an augmented Lagrangian (Nocedal and Wright, 1999, Equation 17.36), and adds a penalty in the objective for any vector $\boldsymbol{z}$ such that $\boldsymbol{A}\boldsymbol{z} \neq \boldsymbol{b}$.

Inequality case. When the additional constraints are inequalities, we have $S = \{\boldsymbol{u} | \boldsymbol{u} \leq \boldsymbol{b}\}$. In that case, we demonstrate in Appendix D that the smoothed indicator function can be expressed as:

$$\delta^{**}_{S,\beta}(\boldsymbol{A}\boldsymbol{z}) = \frac{1}{2\beta}\|[\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}]_+\|_2^2$$

where $[\cdot]_+$ denotes the Euclidean projection into the non-negative orthant, *i.e.* negative values are clipped. It introduces a penalty in the objective for any vector $\boldsymbol{z}$ such that $\boldsymbol{A}\boldsymbol{z} > \boldsymbol{b}$. This is also known as the Courant-Beltrami penalty.

4.3 Structure of maximum weight

Remember that our lower bound to the objective function requires computing the structure of maximum weight. In this section, we introduce two algorithms to do so. The first one relies on the optimization framework presented in Section 4.2. In the second one, we reduce our problem to MAP inference in a factor graph.

4.3.1 Structure of maximum weight via the conditional gradient method

Our optimization problem is defined as:

$$\begin{aligned} \max_{\boldsymbol{x},\boldsymbol{y}} \quad & \mu^\top \boldsymbol{x} + \phi^\top \boldsymbol{y} \\ \text{s.t.} \quad & \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^*_{\text{(relaxed)}} \\ & \sum_{e \in E} x_{v,e} \leq 1 \qquad \forall v \in V \end{aligned}$$

Using the framework of Section 4.2, we can solve the linear relaxation of this problem where the inequality constraints are introduced as a smoothed penalty in the objective instead. We only require an efficient LMO to use as a subroutine in the conditional gradient method. Given a sentence $w = (w_1, \ldots, w_n)$, remember that $G = \langle V, A \rangle$ is a complete graph constructed over w and $G' = \langle V', A' \rangle$ is the semantic graph where each vertex $v' \in V'$ is tagged with a concept $e \in E$ and each arc $a' \in A'$ is labeled with a label $l \in L$.

Linear oracle for arborescences. In this paragraph, we assume that G' is an arborescence. As the set $\mathcal{C}^*_{\text{(relaxed)}}$ only requires each vertex $v' \in V'$ to be anchored on a vertex $v \in V$, this problem is tractable via dynamic programming. Indeed, for a given vertex $v' \in V'$, assume that we know the anchoring of maximum weight of the sub-arborescence rooted at each of its descendents u' for each possible anchor $u \in V$ for u'. The maximum weight of the anchoring of the sub-arborescence rooted at v' where v' is anchored on $v \in V$ is the weight of tagging v with e plus selecting for each descendent u' the anchoring such that the sum of the weights of the sub-arborescence and the arc $v' \to u'$ is maximized. To this end, we can simply visit the set of vertices V' in reverse topological order.

> **Definition 4.13: Topological order**
>
> Let $G = \langle V, A \rangle$ be a directed acyclic graph. A topological order of the vertices V is an order such that for any arc $u \to v \in A$, u must appear before v. In a reverse topological order, u must appear after v.

We give a pseudo-code in Algorithm 2. The best anchoring can then be retrieved via

Algorithm 2 Unconstrained anchoring of maximum weight of G' on a graph G

function DynProgAnchoring(G, G')
  for $u' \in V'$ in reverse topological order **do**
    e is the concept associated to u'
    for $u \in V$ **do**
      Weight$[u', u] \leftarrow \mu_{u,e}$
      for $v' \in \{v'' \in V' \mid u' \to v'' \in A'\}$ **do** ▷ Best anchoring for each descendent
        l is the label associated to $u' \to v'$
        Weight$[u', u] \leftarrow$ Weight$[u', u] + \max_{v \in V}$ (Weight$[v', v] + \phi_{u \to v, l}$)
  return $\max_{u \in V}$ Weight$[r', u]$ ▷ Where $r' \in V'$ is the root of G'

back-pointers. This algorithm requires iterating over every possible anchoring of u' and v' for each arc $u' \to v' \in A'$. As there are $|V'| - 1$ arcs in A' and at most n vertices in V' and in V, this algorithm has a $\mathcal{O}(n^3)$ time complexity.

Linear oracle for non-arborescence graphs. In this paragraph, we assume that G' is not an arborescence. We consider the equivalent undirected graph $H' = \langle V', E' \rangle$ where V' is the same set of vertices and E' is a set of edges defined such that for each arc $u' \to v' \in A'$, there is an edge $u' - v' \in E'$. If H' is acyclic, we can adapt Algorithm 2 by arbitrarily choosing a vertex $v' \in V'$ as a root. Then, we can construct a visiting order by iteratively adding vertices that are not yet in the order but are adjacent to vertices that are.

In the case where there are cycles in H', we propose an approach inspired by algorithms for efficient inference in graphical models, notably the Junction Tree Algorithm (Barber, 2011, Section 6).

> **Definition 4.14: Clique**
>
> Let $G = \langle V, E \rangle$ be an undirected graph. A clique C is a subset of V such that every two distinct nodes in C are adjacent, *i.e.* $\forall u, v \in C : u - v \in E$. If there is no clique C' such that $C \subset C'$, then C is a maximal clique.

> **Definition 4.15: Clique graph**
>
> Let $G = \langle V, E \rangle$ be an undirected graph. A clique graph of G is a graph $G' = \langle V', E' \rangle$ such that each node $v' \in V'$ represents a maximal clique C in G. For every edge

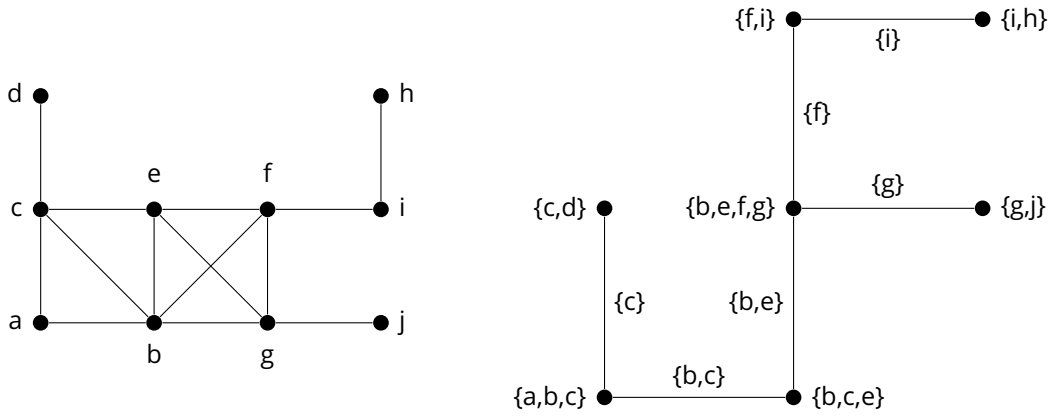

Figure 4.4: **(left)** A graph that contains cliques with more than two vertices. We use letters to identify the vertices. **(right)** A junction tree corresponding to the graph on the left. Beside each node, we indicate the clique from the original graph that it represents. For each edge, we indicate the separator between the two adjacent nodes.

> $u' - v' \in E'$ such that u' and v' represent two cliques C and C' respectively, the edge is labeled with $C \cap C'$ which is known as the separator.

> **Definition 4.16: Junction Tree**
>
> A junction tree is a clique graph $G' = \langle V', E' \rangle$ such that G' is a tree and that, for any two nodes u' and v' in V' representing two cliques C and C' respectively, the separator of every edge on the path between u' and v' contains $C \cap C'$.

We give an example of a graph with its corresponding junction tree in Figure 4.4. Given our undirected graph H', we compute its junction tree. Using this junction tree, we can compute a visiting order over cliques instead of vertices. We can then adapt Algorithm 2 to compute the anchoring of maximum weight by computing the possible anchorings for each clique jointly.

Algorithm complexity. Each node in the junction tree represents a clique containing m vertices. As each vertex in V' is anchored on a vertex among n possible ones in V, there are n^m possible anchorings that must be computed for a clique. As a junction tree can have at most $|V'|$ vertices, computing the anchoring of maximum weight has a $O(n^k)$ time complexity where n is the length of the sentence and k is the size of the largest clique in H.

Dataset	Split	Time complexity			
		$< O(n^3)$	$O(n^3)$	$O(n^4)$	$O(n^5)$
GeoQuery	IID	0.2%	99.8%	-	-
	Template	-	100%	-	-
	Length	0.2%	99.8%	-	-
Scan-SP	IID	0.1%	99.9%	-	-
	Right	0.1%	99.9%	-	-
	AroundRight	0.1%	99.9%	-	-
Clevr	IID	< 0.1%	> 99.9%	-	-
	Closure	< 0.1%	> 99.9%	-	-
COGS	IID	0.6%	93.9%	5.5%	-
	Generalisation	0.6%	93.9%	5.5%	-
CFQ	IID	-	76.1%	23.9%	-
	MCD1	-	72.7%	27.3%	-
	MCD2	-	74.0%	26.0%	< 0.01%
	MCD3	-	72.4%	27.6%	< 0.01%

Table 4.1: Proportions of semantic graphs with respect to the time complexity of the Linear Minimization Oracle in the training set of each data split.

In the datasets considered in this thesis, the semantic graphs possess small cliques in practice. For each split of each dataset, we report in Table 4.1 the distribution of time complexities over the sentences in the train set. GeoQuery, Scan-SP and Clevr only contain arborescences, thus the time complexity is $O(n^3)$ for every structure with at least three vertices in these datasets. The complexity of the LMO is linear or quadratic if the semantic graph contains exactly one or two vertices respectively. COGS and CFQ both contain semantic graphs that are not arborescences. For these two datasets, we thus require the variant based on the Junction Tree Algorithm. However, we observe in Table 4.1 that graphs for which the anchoring of maximum weight can be computed with a cubic complexity cover more than 70% of the datasets. Structures for which the anchoring can be computed with a $O(n^4)$ time complexity cover the remainder of these datasets with the exception of one sentence in the MCD2 and MCD3 splits of CFQ. Thus, handling graphs that are not arborescences does not lead to significantly worse runtimes.

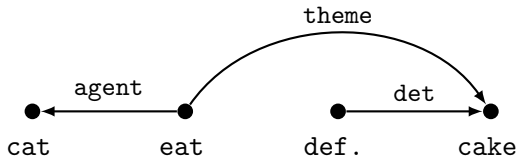

Figure 4.5: The semantic graph corresponding to the sentence "A cat ate the cake" in the COGS dataset.

4.3.2 Structure of maximum weight via a factor graph

To compute the structure of maximum weight, we also propose to reduce our problem to MAP inference in a factor graph. In this section, we explain how to construct this factor graph. An advantage of this approach is its simplicity of use as it is not dependent on the structure of the semantic graph like the previous approach.

Factor graph. For clarity reasons, we denote $v_i \in V$ the vertex corresponding to the word w_i. For each vertex $v' \in V'$ tagged with the concept e, we define a random variable (RV) $X_{v'}$ taking values in $\{1, \ldots, n\}$. When $X_{v'}$ takes the value i, it represents the anchoring of v' on the vertex $v_i \in V$. We introduce a unary factor corresponding to tagging weights, *i.e.* anchoring v' on a vertex v_i induces the weight $\mu_{v_i,e}$. For each arc $u' \to v' \in A'$ labeled with l, we introduce a binary factor corresponding to labeling weights, *i.e.* anchoring u' on a vertex v_i and v' on a vertex v_j induces the weight $\phi_{v_i \to v_j,l}$. Finally, there is a global factor acting as an indicator function, that forbids RVs assignments where different concept instances are aligned with the same word. We use AD3 (Martins et al., 2011) for MAP inference in this factor graph.

> **Example 4.17: Factor graph construction**
>
> Given the sentence "A cat ate the cake" in the COGS dataset, its corresponding semantic graph is shown in Figure 4.5.
>
> We construct the factor graph that is used to compute the anchoring of maximum weight for this semantic graph in Figure 4.6. For each of the four vertices, we introduce a random variable denoted A_i with a unary factor denoted u_i corresponding to its concept, *i.e.* `cat`, `eat`, `the` and `cake`. For each arc $u' \to v'$ in the semantic graph, we introduce a binary factor denoted b_{ij} between the random variables A_i and A_j

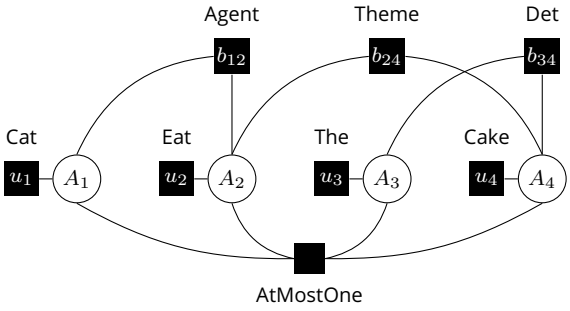

Figure 4.6: Factor graph constructed to compute the anchoring of maximum weight of the semantic graph shown in Figure 4.5 for the sentence "A cat ate the cake". A detailed description is given in Example 4.17.

A_i	u_i
1	$\mu_{v_1,e}$
2	$\mu_{v_2,e}$
3	$\mu_{v_3,e}$
4	$\mu_{v_4,e}$
5	$\mu_{v_5,e}$

$A_i \setminus A_j$	1	2	3	4	5
1	$-\infty$	$\phi_{v_1\to v_2,l}$	$\phi_{v_1\to v_3,l}$	$\phi_{v_1\to v_4,l}$	$\phi_{v_1\to v_5,l}$
2	$\phi_{v_2\to v_1,l}$	$-\infty$	$\phi_{v_2\to v_3,l}$	$\phi_{v_2\to v_4,l}$	$\phi_{v_2\to v_5,l}$
3	$\phi_{v_3\to v_1,l}$	$\phi_{v_3\to v_2,l}$	$-\infty$	$\phi_{v_3\to v_4,l}$	$\phi_{v_3\to v_5,l}$
4	$\phi_{v_4\to v_1,l}$	$\phi_{v_4\to v_2,l}$	$\phi_{v_4\to v_3,l}$	$-\infty$	$\phi_{v_4\to v_5,l}$
5	$\phi_{v_5\to v_1,l}$	$\phi_{v_5\to v_2,l}$	$\phi_{v_5\to v_3,l}$	$\phi_{v_5\to v_4,l}$	$-\infty$

Figure 4.7: **(left)** Weights induced by a unary factor u_i in the factor graph given the value taken by the random variable A_i. We denote e the concept corresponding to this unary factor. **(right)** Weights induced by a binary factor b_{ij} in the factor graph given the values taken by A_i and A_j. We denote l the label corresponding to this binary factor.

> where A_i and A_j correspond to u' and v' respectively. In this case, we have three binary factors for the labels `agent`, `theme` and `det`. Finally, we introduce the AtMostOne global factor ensuring that each vertex in the semantic graph will be anchored on a different word in the sentence.
>
> In Figure 4.7, we represent the weights induced by each unary and binary factor in our factor graph.

4.4 Conclusion

In this chapter, we focused on two topics. The first one is the introduction of a combinatorial optimization framework with constraint relaxation. It relies on the conditional gradient algorithm to solve the linear relaxation of optimization problems if it is convex. Constraints that prevent the existence of an efficient oracle are relaxed and introduced as quadratic penalties in the objective instead. This framework is appealing as it can be applied to any optimization problem such that its linear relaxation is convex. We rely on

this framework to compute our training objective and in the inference algorithm of our contribution presented in Chapter 5.

The second focus of this chapter is a mathematically motivated objective function to train graph-based semantic parsers in a "weakly-supervised" setting. We demonstrated that the objective function in this setting is composed of two intractable LSE functions, one over $\mathcal{C}^*$, the set of structures that induce the gold semantic graph, and the other over $\mathcal{C}$, the set of all valid structures. We proposed an upper bound to the LSE function over $\mathcal{C}$ by decomposing it over each word and each pair of words. For the LSE function over $\mathcal{C}^*$, we proved that considering only the structure of maximum weight yields a lower bound. As computing this structure is NP-hard, we proposed two approximate algorithms. As mentionned above, the first one relies on the combinatorial optimization framework presented in this chapter and an efficient oracle. In the second one, we reduce our problem to MAP inference in a factor graph.

Chapter 5

Graph-based reentrancy-free semantic parsing

In Chapter 2, we explained that the advances of sequence-to-sequence architectures for machine translation (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015) made them appealing for semantic parsing (Jia and Liang, 2016; Dong and Lapata, 2016; Wang et al., 2020). However, the recent focus on compositional generalization highlighted that these models tend to fail when composition is required (Lake and Baroni, 2018; Finegan-Dollak et al., 2018; Keysers et al., 2020). As such, there has been an interest in revisiting more traditional structured prediction methods, notably predicting partial programs over short utterances and composing them (Pasupat et al., 2019; Herzig and Berant, 2021).

Our novel contribution is a graph-based semantic parser that predicts the entire semantic graph jointly. In Section 5.1, we motivate our approach by presenting the limitations of recent work relying on a span-based approach. Then, we introduce our novel graph-based approach that bypasses these limitations in Section 5.2. In Section 5.3, we first prove that MAP inference is NP-hard for this approach and present an integer linear programming formulation for the problem. We propose an approximate solver based on the conditional gradient method presented in Chapter 4. Finally, we evaluate our approach on GeoQuery, Clevr and SCAN in Section 5.4 and observe that it outperforms comparable baselines both on IID splits and splits that test for compositional generalization. Part of the contributions in this chapter were published in Petit and Corro (2023).

5.1 Motivation

In Section 2.3.2, we introduced span-based approaches for semantic parsing (Pasupat et al., 2019; Herzig and Berant, 2021). To improve compositional generalization, these approaches drew inspiration from older grammar-based approaches by predicting partial semantic programs over short utterances and composing them. During inference, SpanBasedSP (Herzig and Berant, 2021) relies on standard span-based decoding algorithms (Hall et al., 2014; Stern et al., 2017; Corro, 2020) with additional well-formedness constraints from the semantic formalism. Given a weighting function, MAP inference is a polynomial time problem that can be solved via a variant of the CYK algorithm (Kasami, 1965; Younger, 1967; Cocke, 1970). Experimentally, SpanBasedSP outperforms sequence-to-sequence models in terms of compositional generalization.

The complexity of MAP inference for phrase structure parsing is directly impacted by the considered search space (Kallmeyer, 2010). Importantly, (ill-nested) discontinuous phrase structure parsing is known to be NP-hard, even with a bounded block-degree (Satta, 1992). Herzig and Berant (2021) explore two restricted inference algorithms, both of which have a cubic time complexity with respect to the input length. The first one only considers continuous phrase structures, *i.e.* derived trees that could have been generated by a context-free grammar, and the second one also considers a specific type of discontinuities, see Corro (2020, Section 3.6). Both algorithms fail to cover the full set of phrase structures observed in semantic treebanks as illustrated in Figure 5.1.

In this chapter, we propose to reduce semantic parsing without reentrancy to a bilexical dependency parsing problem. Forbidding reentrancies means that a given predicate or entity cannot be used as an argument for two different predicates, As such, we tackle the same semantic content as aforementioned previous work but using a different mathematical representation (Rambow, 2010). We identify two main benefits to our approach: first, as we allow crossing arcs, all datasets are guaranteed to be fully covered. Secondly, it allows us to rely on optimization methods to tackle the inference of our graph-based approach. More specifically, we will prove that MAP inference in our setting is equivalent to the maximum generalized spanning arborescence problem (Myung et al., 1995) with supplementary constraints. Although this problem is NP-hard, we propose an

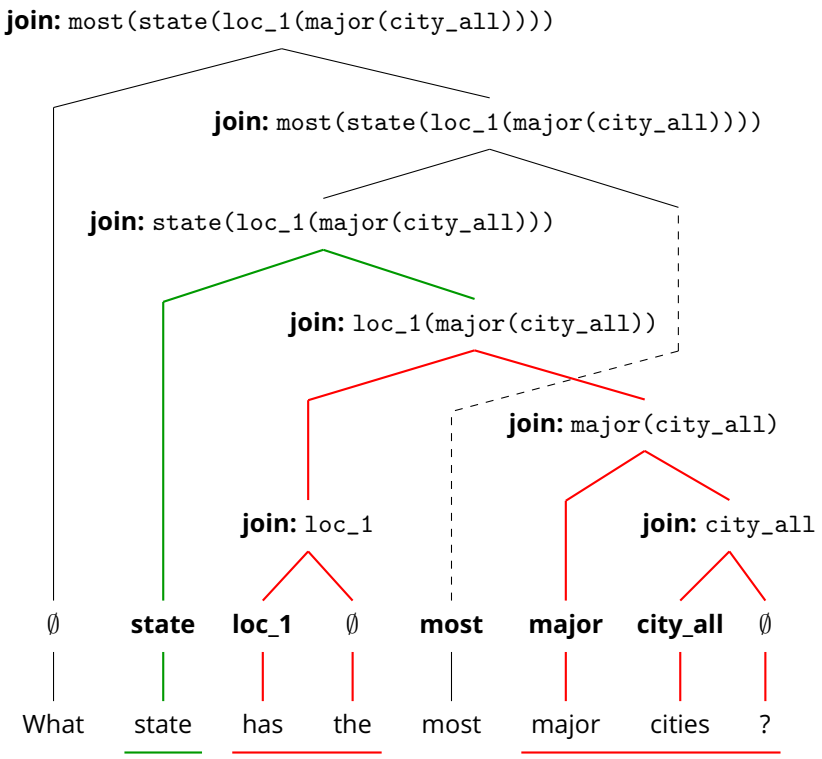

Figure 5.1: Example of a semantic phrase structure from GeoQuery. This structure is outside of the search space of SpanBasedSP (Herzig and Berant, 2021) as the constituent in red is discontinuous and also has a discontinuous parent (in red and green).

optimization algorithm that solves a linear relaxation of the problem.

5.2 Well-formed graph-based semantic parsing

5.2.1 Graph notations and definitions

We first introduce the graph notations and definitions that are relevant to the work presented in this chapter. Let $G = \langle V, A \rangle$ be a directed graph with V the set of vertices and $A \subseteq V \times V$ the set of arcs. For any subset of vertices $U \subseteq V$, we denote $\sigma_G^+(U)$ (respectively $\sigma_G^-(U)$) the set of arcs leaving one vertex of U and entering one vertex of $V \setminus U$ (respectively leaving one vertex of $V \setminus U$ and entering one vertex of U) in the graph G.

> **Definition 5.1: Cover set**
>
> Let $G = \langle V, A \rangle$ be a directed graph. Let $B \subseteq A$ be a subset of arcs. The cover set of B denoted $V[B]$ is the set of vertices that appear as an extremity of at least one arc

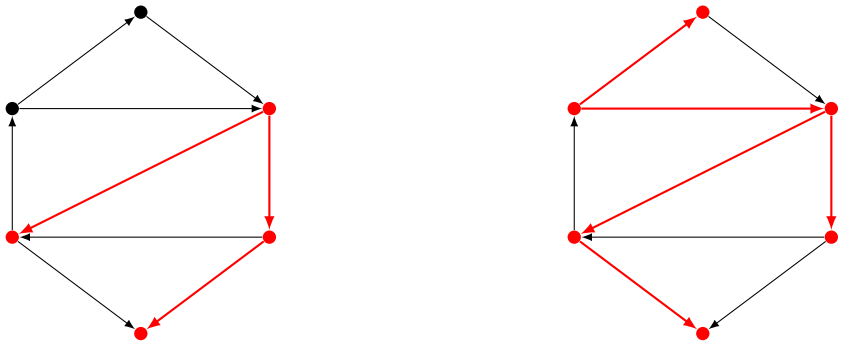

Figure 5.2: Illustrations of arborescences (in red) in a directed graph. **(left)** The arborescence does not cover every vertex in the graph. **(right)** The arborescence covers every vertex, it is spanning.

in B. We can define it formally as

$$V[B] = \{v \mid \exists u : u \to v \in B \vee v \to u \in B\}$$

Definition 5.2: Arborescence and spanning arborescence

Let $G = \langle V, A \rangle$ be a directed graph. G is an arborescence rooted at $u \in V$ if and only if it contains $|V| - 1$ arcs and there is a directed path from u to each vertex in V. Let $B \subseteq A$ be a set of arcs such that $G' = \langle V[B], B \rangle$ is an arborescence. G' is a spanning arborescence of G if and only if the cover set $V[B] = V$.

We give examples of an arborescence and a spanning arborescence in Figure 5.2. In this chapter, we assume that the root of an arborescence is always a vertex $0 \in V$.

Definition 5.3: Cluster and partition

Let $G = \langle V, A \rangle$ be a graph. A cluster is a subset of vertices $U \subseteq V$. Let $\pi = \{V_0, \ldots, V_n\}$ be a set of $n + 1$ clusters in V. π is a partition of V if and only if we have

$$\bigcup_{V_i \in \pi} V_i = V$$

and

$$\forall V_i, V_j \in \pi : V_i \cap V_j = \emptyset$$

In other words, each vertex $v \in V$ belongs to exactly one cluster $V_i \in \pi$.

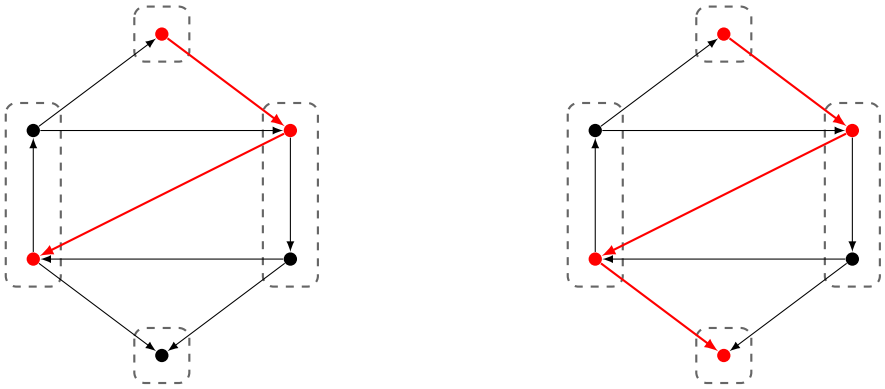

Figure 5.3: Illustrations of generalized arborescences (in red) in a directed graph. The clusters are indicated by the dashed lines. **(left)** The generalized arborescence does not cover a vertex in every cluster, it is not-necessarily spanning. **(right)** The generalized arborescence covers a vertex in every cluster, it is spanning.

Definition 5.4: Generalized arborescence

Let $G = \langle V, A \rangle$ be a directed graph and π be a partition of V as presented in Definition 5.3. Let $G' = \langle V[B], B \rangle$ be an arborescence of G as introduced in Definition 5.2. G' is a generalized not-necessarily spanning arborescence on the partition π of G if and only if $V[B]$ contains at most one vertex per cluster $V_i \in \pi$. If it contains exactly one vertex per cluster, it is a generalized spanning arborescence.

We give examples of generalized spanning and not-necessarily spanning arborescences in Figure 5.3.

Definition 5.5: Contracted graph

Let $G = \langle V, A \rangle$ be a directed graph. Let $W \subseteq V$ be a set of vertices. Contracting W consists in replacing in G the set W by a new vertex $w \notin V$, each arc $u \to v \in \sigma^-(W)$ by an arc $u \to w$ and each arc $u \to v \in \sigma^+(W)$ by an arc $w \to v$.

Given $\pi = \{V_0, \ldots, V_n\}$ a partition of G, the contracted graph is the graph where each cluster $V_i \in \pi$ has been contracted.

We give an example of a contracted graph in Figure 5.4. Note that while contracting a graph may introduce parallel arcs, this is not an issue in practice for our work.

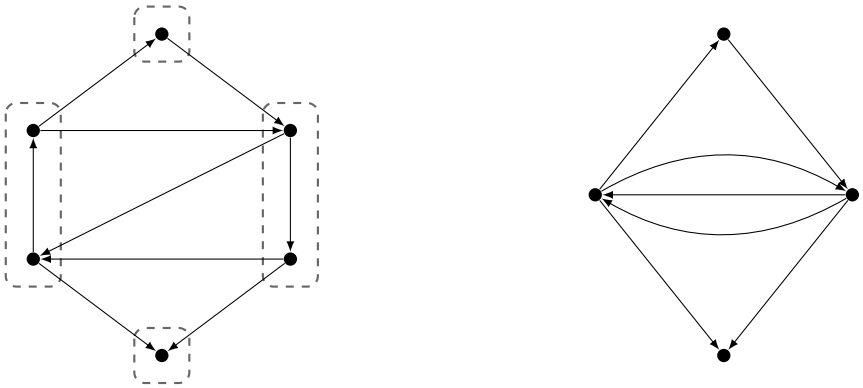

Figure 5.4: **(left)** A directed graph where clusters are represented by the dashed lines. **(right)** The resulting graph when each cluster is contracted.

5.2.2 Semantic grammar and graph

The semantic programs we focus on in this chapter take the form of a functional language, *i.e.* a representation where each predicate is a function that takes other predicates or entities as arguments. The semantic language is typed in the same sense than in "typed programming languages". For example, in GeoQuery, the predicate `capital_2` expects an argument of type `city` and returns an object of type `state`. We give a complete description of the typing systems for GeoQuery, Clevr and SCAN in Appendices A.2, C.2 and B.2 respectively. In these datasets, the typing system disambiguates the position of arguments in a function: for a given function, either all arguments are of the same type or the order of arguments is unimportant. An example is the predicate `intersection_river` in GeoQuery that takes two arguments of type `river`, but the result of the execution is unchanged if the arguments are swapped. There are a few corner cases like `exclude_river`, for which we simply assume arguments are in the same order as they appear in the input sentence.

Definition 5.6: Semantic grammar

We define a semantic grammar as $\mathcal{G} = \langle E, T, f_{\text{Type}}, f_{\text{Args}} \rangle$ where:

- E is the set of predicates and entities, which we refer to as the set of concepts. We assume that Root $\notin E$ is a special concept that is used for parsing.
- T is the set of types.

- $f_{\text{Type}} : E \to T$ is a typing function that assigns a type to each concept.
- $f_{\text{Args}} : E \times T \to \mathbb{N}$ is a valency function that assigns the number of expected arguments of a given type to each concept.

Formally, we define the set of valid semantic programs as the set of programs that can be produced with a semantic grammar defined as in Definition 5.6. Note that a concept $e \in E$ is an entity if and only if $\forall t \in T : f_{\text{Args}}(e, t) = 0$. Otherwise, it is a predicate.

A semantic program in a functional language can be equivalently represented as a semantic graph where instances of concepts are represented as vertices and where arcs identify arguments of predicates. The conversion for GeoQuery, SCAN and Clevr is detailed in Section 3.5.

Definition 5.7: Well-formed semantic graph

A semantic graph is defined as a labeled graph $G = \langle V, A, f_{\text{Label}} \rangle$ where the function $f_{\text{Label}} : V \to E$ assigns a concept to each vertex. G is well-formed with respect to a semantic grammar $\mathcal{G}$ if and only if G is an arborescence and the valency and type constraints are satisfied, *i.e.* we have:

$$f_{\text{Args}}(f_{\text{Label}}(u), t) = |\sigma_G^+(\{u\}, t)| \qquad \forall u \in V, t \in T$$

where:

$$\sigma_G^+(\{u\}, t) = \left\{ \begin{array}{l} u \to v \in \sigma_G^+(\{u\}) \\ \text{s.t. } f_{\text{Type}}(f_{\text{Label}}(v)) = t \end{array} \right\}.$$

5.2.3 Problem reduction

In our setting, semantic parsing is a joint sentence tagging and dependency parsing problem (Bohnet and Nivre, 2012; Li et al., 2011; Corro et al., 2017): each content word (*i.e.* words that convey a semantic meaning) must be tagged with a predicate or an entity, and dependencies between content words identify arguments of predicates. However, our semantic parsing setting differs from standard syntactic analysis in two ways:

- The resulting structure is not necessarily spanning as there are words that must

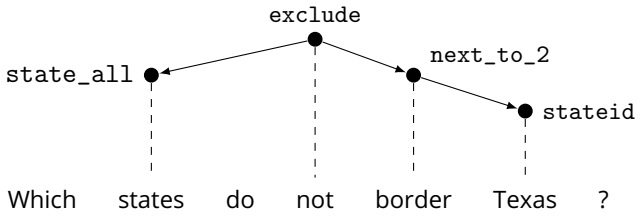

Figure 5.5: Semantic graph (solid arcs) corresponding to the sentence "Which states do not border Texas ?" in the GeoQuery dataset. The dashed lines represent the anchors of the concepts (note that this information is not available in the dataset).

not be tagged and that do not have any incident dependency. Those words are not known in advance and must be identified jointly with the rest of the structure.

- The dependency structure is highly constrained by the typing mechanism as the predicted structure must be a well-formed semantic graph.

Nevertheless, similarly to aforementioned works, our parser is graph-based, *i.e.* for a given input we build a (complete) directed graph and decoding is then reduced to computing a constrained subgraph of maximum weight.

Clustered labeled graph. Given a sentence $w = (w_1, ..., w_n)$ with n words and a semantic grammar $\mathcal{G}$, we construct a clustered labeled graph $G = \langle V, A, \pi, \bar{f}_{\text{Label}} \rangle$ as follows. The partition $\pi = \{V_0, ..., V_n\}$ contains $n+1$ clusters, where V_0 is a root cluster and each cluster V_i, $i \neq 0$, is associated to word w_i. The root cluster $V_0 = \{0\}$ contains a single vertex that will be used as the root and every other cluster contains $|E|$ vertices. The extended labeling function $\bar{f}_{\text{Label}} : V \to E \cup \{\text{Root}\}$ assigns a concept in E to each vertex $v \in V \setminus \{0\}$ and Root to vertex 0. Distinct vertices in a cluster V_i cannot have the same concept, *i.e.* $\forall u, v \in V_i : u \neq v \implies \bar{f}_{\text{Label}}(u) \neq \bar{f}_{\text{Label}}(v)$.

Let $B \subseteq A$ be a subset of arcs. The graph $G' = \langle V[B], B \rangle$ defines a 0-rooted generalized valency-constrained not-necessarily-spanning arborescence (GVCNNSA) if and only if it is a generalized arborescence of G, there is exactly one arc $0 \to u$ leaving 0 and the subarborescence rooted at u is a well-formed semantic graph with respect to the grammar $\mathcal{G}$. As such, there is a one-to-one correspondence between a semantic graph anchored on the sentence w and a GVCNNSA in the weighted graph G.

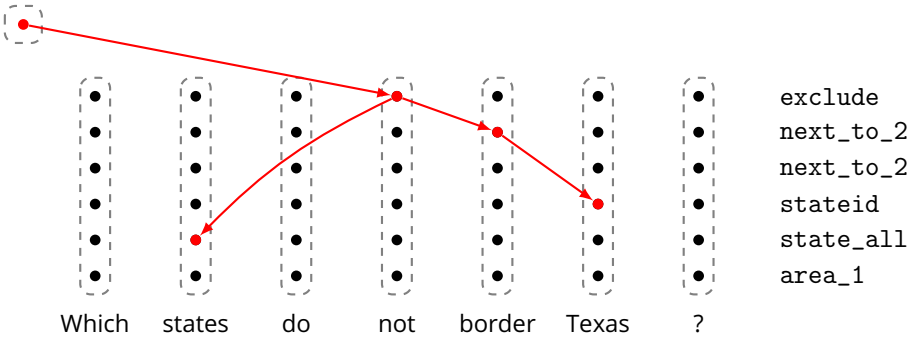

Figure 5.6: A labeled clustered graph with the generalized valency-constrained not-necessarily-spanning arborescence (red arcs) corresponding to the semantic graph in Figure 5.5. For clarity reasons, we do not represent the other arcs in the graph. The root is the top left vertex.

> **Example 5.8: Clustered labeled graph construction**
>
> In the GeoQuery dataset, the sentence "Which states do not border Texas ?" is mapped to the program "`exclude(state_all, next_to_2(stateid(texas)))`". We show the corresponding semantic graph in Figure 5.5. The corresponding clustered labeled graph is presented in Figure 5.6. There is a cluster corresponding to each word in the sentence as well as an additional root cluster. For compactness reasons, we only included a few concepts from GeoQuery in the figure. The GVCNNSA corresponding to the semantic graph is shown in red.

For a given sentence w, our aim is to find the semantic graph that most likely corresponds to it. Thus, after building the clustered labeled graph G as explained above, we use a neural network that will be described in Section 5.4 to produce a vector of weights $\boldsymbol{\mu} \in \mathbb{R}^{|V|}$ indexed by the set of vertices V and a vector of weights $\boldsymbol{\phi} \in \mathbb{R}^{|A|}$ indexed by the set of arcs A. Given these weights, our graph-based semantic parsing task is reduced to an optimization problem: the maximum generalized valency-constrained not-necessarily-spanning arborescence (MGVCNNSA) in the graph G.

5.3 Mathematical formulation and resolution

In the previous section, we have cast our semantic parsing problem as an optimization problem. In this section, we first prove that this problem is NP-hard. Then, we present

an integer linear programming (ILP) formulation for our problem. Finally, we highlight which constraints are problematic and propose a way to solve the linear relaxation of our problem by relying on the conditional gradient method, similarly to Chapter 4.

5.3.1 NP-hardness of the MGVCNNSA problem

> **Theorem 5.9: NP-hardness of the MGVCNNSA**
>
> The MGVCNNSA problem is NP-hard.

Proof. The maximum not-necessarily-spanning arborescence (MNNSA) problem is known to be NP-hard (Rao and Sridharan, 2002; Duhamel et al., 2008). We prove Theorem 5.9 by reducing it to the MGVCNNSA problem.

Let $G = \langle V, A, \boldsymbol{\psi} \rangle$ be a weighted graph where $V = \{0, ..., n\}$ and $\boldsymbol{\psi} \in \mathbb{R}^{|A|}$ are arc weights. The MNNSA problem aims to compute the subset of arcs $B \subseteq A$ such that $\langle V[B], B \rangle$ is an arborescence of maximum weight, where its weight is defined as $\sum_{a \in B} \psi_a$.

Let $\mathcal{G} = \langle E, T, f_{\mathsf{Type}}, f_{\mathsf{Args}} \rangle$ be a grammar such that $E = \{0, ..., n-1\}$, $T = \{t\}$ and $\forall e \in E : f_{\mathsf{Type}}(e) = t \wedge f_{\mathsf{Args}}(e, t) = e$. Intuitively, a concept $e \in E$ is associated to vertices that require exactly e outgoing arcs.

We construct a clustered labeled weighted graph $G' = \langle V', A', \pi, f_{\mathsf{Label}}, \boldsymbol{\psi}' \rangle$ as follows. $\pi = \{V'_0, ..., V'_n\}$ is a partition of V' such that each cluster $V'_i \in \pi$ contains $n-1$ vertices and represents the vertex $i \in V$. The labeling function f_{Label} assigns a different concept to each vertex in a cluster, *i.e.* $\forall V'_i \in \pi, \forall u', v' \in V'_i : u' \neq v' \Rightarrow l(u') \neq l(v')$. The set of arcs is defined as $A' = \{u' \to v' | \exists i \to j \in A \text{ s.t. } u' \in V'_i \wedge v' \in V'_j\}$. The weight vector $\boldsymbol{\psi}' \in \mathbb{R}^{|A'|}$ is such that $\forall u' \to v' \in A' : u' \in V'_u \wedge v' \in V'_v \Rightarrow \psi'_{u' \to v'} = \psi_{u \to v}$.

As such, there is a polynomial one-to-one correspondence between solutions of the MNNSA problem on graph G and solutions of the MGVCNNSA problem on graph G'. Thus, the MGVCNNSA problem is also NP-hard. □

5.3.2 Mathematical program

Our graph-based approach to semantic parsing has allowed us to prove the intrinsic hardness of the problem. We now follow previous work on graph-based parsing (Martins et al.,

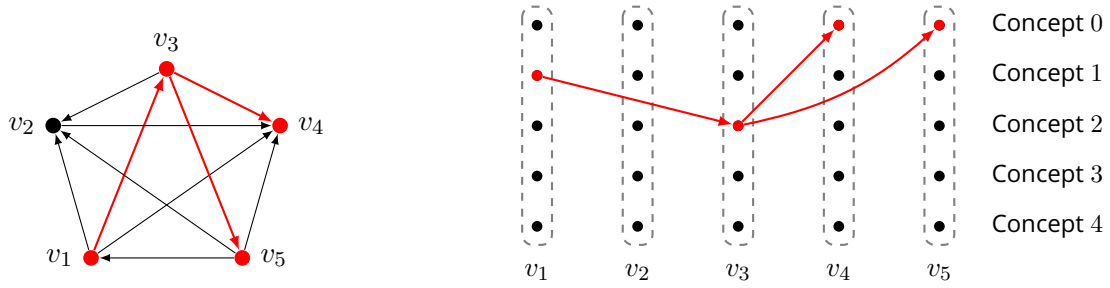

Figure 5.7: Illustration of the reduction used in the proof of Theorem 5.9. **(left)** A graph with its maximum not-necessarily-spanning arborescence in red. **(right)** The corresponding clustered graph and MGVCNNSA in red. We do not represent the other arcs present in the clustered graph for clarity reasons.

2009; Koo et al., 2010) by proposing an integer linear programming formulation in order to compute (approximate) solutions.

Extended graph. Remember that in the joint tagging and dependency parsing interpretation of the semantic parsing problem, the resulting structure is not-necessarily-spanning, meaning that some words may not be tagged. In order to rely on well-known algorithms for computing spanning arborescences, we first introduce the notion of extended graph. Given a clustered labeled graph $G = \langle V, A, \pi, \bar{f}_{\text{Label}} \rangle$, we construct an extended graph $\overline{G} = \langle \overline{V}, \overline{A}, \overline{\pi}, \bar{f}_{\text{Label}} \rangle$ containing n additional vertices $\{\overline{1}, ..., \overline{n}\}$ that are distributed along clusters, *i.e.* $\overline{\pi} = \{V_0, V_1 \cup \{\overline{1}\}, ..., V_n \cup \{\overline{n}\}\}$, and arcs from the root to these extra vertices, *i.e.* $\overline{A} = A \cup \{0 \rightarrow \bar{i} | 1 \leq i \leq n\}$. Note that the labeling function is unchanged as there is no need for types for the additional vertices in $\overline{V} \setminus V$. Let $B \subseteq A$ be a subset of arcs such that $\langle V[B], B \rangle$ is a generalized not-necessarily-spanning arborescence on G. Let $\overline{B} \subseteq \overline{A}$ be a subset of arcs defined as $\overline{B} = B \cup \{0 \rightarrow \bar{i} | \sigma^{-}_{\langle V[B], B \rangle}(V_i) = \emptyset\}$. $\langle \overline{V}[\overline{B}], \overline{B} \rangle$ is a generalized spanning arborescence on $\overline{G}$ and there is a one-to-one correspondence between generalized not-necessarily spanning arborescences $\langle V[B], B \rangle$ on G and generalized spanning arborescences $\langle \overline{V}[\overline{B}], \overline{B} \rangle$ on $\overline{G}$.

> **Example 5.10: Extended graph construction**
>
> We show the clustered labeled graph corresponding to the sentence "Which states do not border Texas ?" in Figure 5.6. The corresponding extended graph is shown

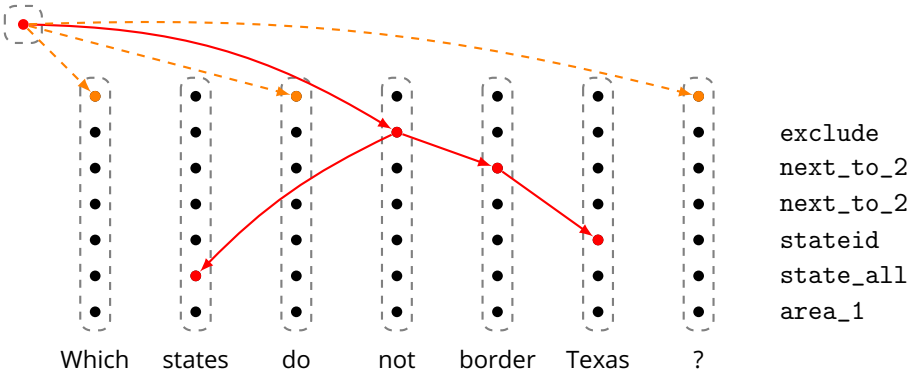

Figure 5.8: The extended graph corresponding to the clustered labeled graph presented in Figure 5.6. A vertex with the $\emptyset$ concept is added to each cluster except the root cluster. Adding the dotted orange arcs and vertices to the GVCNNSA produces a generalized spanning arborescence.

> in Figure 5.8. Each cluster except the root cluster has an additional vertex with the $\emptyset$ concept. As "Which", "do" and "?" are not the anchors of vertices of the semantic graph, the vertices labeled with $\emptyset$ are selected in their respective clusters to produce the corresponding generalized spanning arborescence.

Integer Linear Program. Let $\boldsymbol{x} \in \{0,1\}^{|\overline{V}|}$ and $\boldsymbol{y} \in \{0,1\}^{|\overline{A}|}$ be variable vectors indexed by vertices and arcs such that a vertex $v \in \overline{V}$ (respectively an arc $a \in \overline{A}$) is selected if and only if $x_v = 1$ (respectively $y_a = 1$). The set of 0-rooted generalized valency-constrained spanning arborescences on $\overline{G}$ can be written as the set of variables $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ satisfying the following linear constraints. First, we restrict $\boldsymbol{y}$ to structures that are spanning arborescences over $\overline{G}$ where clusters have been contracted:

$$\sum_{a \in \sigma^-_{\overline{G}}(V_0)} y_a = 0 \tag{5.1}$$

$$\sum_{a \in \sigma^-_{\overline{G}}\left(\bigcup_{\overline{U} \in \overline{\pi'}} \overline{U}\right)} y_a \geq 1 \qquad \forall \overline{\pi'} \subseteq \overline{\pi} \setminus \{V_0\} \tag{5.2}$$

$$\sum_{a \in \sigma^-_{\overline{G}}(\overline{V_i})} y_a = 1 \qquad \forall \overline{V_i} \in \overline{\pi} \setminus \{V_0\} \tag{5.3}$$

Constraint (5.1) ensures that V_0 does not have any incoming arc, *i.e.* it will be the root of the arborescence on the contracted graph. Constraints (5.2) require any subset of clusters to have at least one incoming arc from a cluster not in that subset. As this forbids any subset

of cluster from being isolated, the contracted graph will necessarily be weakly connected. Constraints (5.3) force each cluster except V_0 to have exactly one incoming arc. The set of vectors $\boldsymbol{y}$ that satisfy these three constraints is exactly the set of 0-rooted spanning arborescences on the contracted graph (Schrijver, 2003, Section 52.4).

The root vertex is always selected and other vertices are selected if and only if they have an incoming selected arc:

$$x_0 = 1 \tag{5.4}$$

$$x_u = \sum_{a \in \sigma^-_{\overline{G}}(\{u\})} y_a \qquad \forall u \in \overline{V} \setminus \{0\} \tag{5.5}$$

Note that constraints (5.1)–(5.3) do not force selected arcs to leave from a selected vertex as they operate at the cluster level. This property will be enforced via the following valency constraints:

$$\sum_{u \in V \setminus \{0\}} y_{0 \to u} = 1 \tag{5.6}$$

$$\sum_{a \in \sigma^+_G(\{u\}, t)} y_a = x_u f_{\mathsf{Args}}(l(u), t) \qquad \forall t \in T, u \in V \setminus \{0\} \tag{5.7}$$

Constraint (5.6) forces the root to have exactly one outgoing arc into a vertex $u \in V \setminus \{0\}$ (*i.e.* a vertex that is not part of the extra vertices introduced in the extended graph) that will be the root of the labeled graph. Constraints (5.7) force the selected vertices and arcs to produce a well-formed graph with respect to the grammar $\mathcal{G}$. Note that these constraints are only defined for the vertices in $V \setminus \{0\}$, *i.e.* they are neither defined for the root vertex nor for the extra vertices introduced in the extended graph.

To simplify notations, we introduce the following sets:

$$\mathcal{C}^{(\mathsf{sa})} = \left\{ \begin{array}{l} \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \{0,1\}^{|\overline{V}|} \times \{0,1\}^{|\overline{A}|} \\ \text{s.t. } \boldsymbol{x} \text{ and } \boldsymbol{y} \text{ satisfy (5.1)–(5.5)} \end{array} \right\}$$

$$\mathcal{C}^{(\mathsf{val})} = \left\{ \begin{array}{l} \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \{0,1\}^{|\overline{V}|} \times \{0,1\}^{|\overline{A}|} \\ \text{s.t. } \boldsymbol{x} \text{ and } \boldsymbol{y} \text{ satisfy (5.6)–(5.7)} \end{array} \right\}$$

and $\mathcal{C} = \mathcal{C}^{(\text{sa})} \cap \mathcal{C}^{(\text{val})}$. Given vertex weights $\boldsymbol{\mu} \in \mathbb{R}^{|\overline{V}|}$ and arc weights $\boldsymbol{\phi} \in \mathbb{R}^{|\overline{A}|}$, computing the MGVCNNSA is equivalent to solving the following ILP:

$$\begin{aligned} \max_{\boldsymbol{x},\boldsymbol{y}} \quad & \boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y} \\ \text{s.t.} \quad & \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^{(\text{sa})} \text{ and } \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^{(\text{val})} \end{aligned}$$

5.3.3 Problem resolution

In Chapter 4, we proposed a framework to solve the linear relaxation of NP-hard problems via the conditional gradient method. To apply the same method for the MGVCNNSA problem, we need to identify which constraints prevent us from having an efficient algorithm to compute the optimal solution. Note that the set $\mathcal{C}^{(\text{sa})}$ is the set of spanning arborescences over the contracted graph, hence we can compute the structure of maximum weight in this set as follows:

1. We contract the clustered graph G and assign to each arc in the contracted graph the weight of its corresponding arc plus the weight of its destination vertex in the original graph.

2. We run the Maximum Spanning Arborescence algorithm (MSA, Edmonds, 1967; Tarjan, 1977) on the contracted graph. This algorithm has a $\mathcal{O}(n^2)$ time complexity. Note that, even though the contracted graph may have parallel arcs, it is not an issue in practice as only the arc of maximum weight can appear in a solution of the MSA algorithm.

We illustrate this process in Figure 5.9 and Example 5.11. As such, we propose to relax the constraint $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^{(\text{val})}$ and introduce it as a penalty in the objective instead. We can then solve the linear relaxation of our problem over the set $\text{conv}(\mathcal{C}^{(\text{sa})})$ via the conditional gradient method. Our LMO will be the two-step procedure introduced above over the set $\mathcal{C}^{(\text{sa})}$.

> **Example 5.11: Graph contraction and MSA algorithm**
>
> Illustration of the approximate inference algorithm on the two-word sentence “List

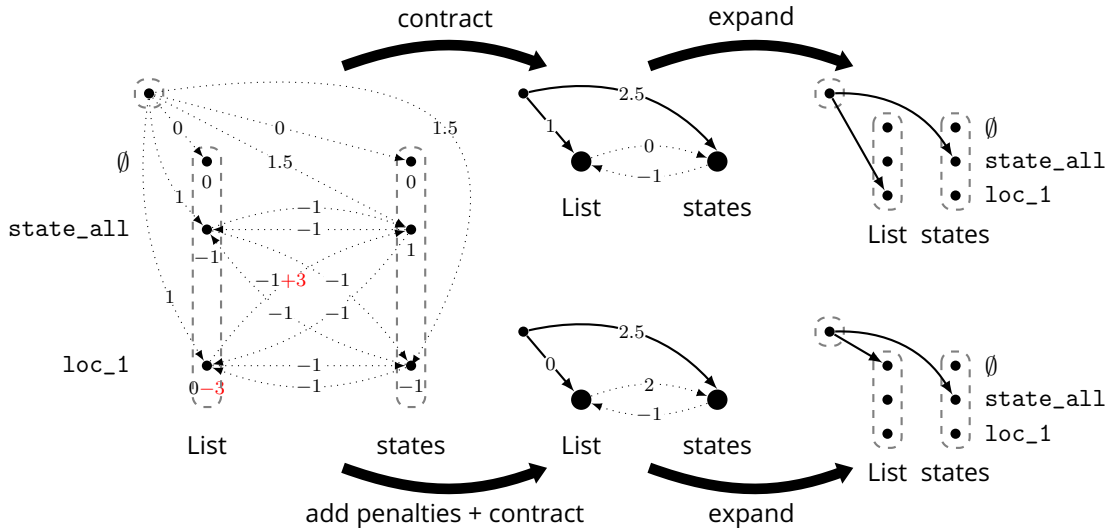

Figure 5.9: Illustration of the approximate inference algorithm on the two-word sentence "List states", where we assume the grammar has one entity `state_all` and one predicate `loc_1` that takes exactly one entity as argument. Detailed explanations are given in Example 5.11.

> states", where we assume the grammar has one entity `state_all` and one predicate `loc_1` that takes exactly one entity as argument. The left graph is the extended graph for the sentence, including vertices and arcs weights (in black). If we ignore constraints (5.6)–(5.7), inference is reduced to computing the MSA on the contracted graph (solid arcs in the middle column). This may lead to solutions that do not satisfy constraints (5.6)–(5.7) on the expanded graph (top example). However, the gradient of the smoothed constraint (5.7) will induce penalties (in red) to vertex and arc scores that will encourage the `loc_1` predicate to either be dropped from the solution or to have an outgoing arc to a `state_all` argument. Computing the MSA on the contracted graph with penalties results in a solution that satisfies constraints (5.6)–(5.7) (bottom example).

5.4 Experiments

5.4.1 Experimental setup

Neural architecture. In our experiments, a neural architecture was used to obtain the weights μ and ϕ. Our baseline architecture is as follows:

- A contextual representation for each word is obtained with an embedding layer of

dimension 200 followed by a bi-LSTM (Hochreiter and Schmidhuber, 1997) with a hidden size of 400.

- This representation is given to a linear projection of dimension 500 followed by a ReLU activation and another projection of dimension $|E|$ to obtain $\boldsymbol{\mu}$.

- The representation is given to another linear projection of dimension 500 followed by a ReLU activation and a bi-affine layer (Dozat and Manning, 2017) to obtain $\boldsymbol{\phi}$.

We apply dropout with a probability of 0.3 on the output of the bi-LSTM and after the ReLU activations. The learning rate is 5×10^{-4} and each batch is composed of 30 sentences. We keep the parameters that obtain the best accuracy on the development set after 25 epochs.

Lexicon. Similarly to previous work (Zettlemoyer and Collins, 2005; Wang et al., 2015; Liang et al., 2017; Herzig and Berant, 2021), we rely on a lexicon to guide our model. For each concept $e \in E$ that is associated with a word w in the lexicon, the score associated to the vertex labeled with e in the cluster corresponding to w is increased by a learned constant λ. This manual work is only done once and is common in semantic parsing. We detail the lexicons used for GeoQuery, Clevr and SCAN in Appendices A.1, C.1 and B.1 respectively.

Training objective. The datasets used in this chapter do not contain the anchoring of the semantic graphs over the sentences. In Chapter 4, we proposed a weakly-supervised loss that relied on computing the anchoring of maximum weight between a complete graph and the semantic graph. For a given semantic graph $G = \langle V, A, f_{\text{Label}} \rangle$ and a clustered labeled graph $G' = \langle V', A', \pi, \bar{f}_{\text{Label}} \rangle$, if we require each vertex $v \in V$ to be anchored on a vertex $v' \in V'$ such that $f_{\text{Label}}(v) = \bar{f}_{\text{Label}}(v')$, computing the anchoring of maximum weight can be reduced to the problem introduced in Chapter 4. Thus, we can use this approach to compute the training objective.

	SCAN		
	IID	Right	AroundRight
Baselines (denotation accuracy)			
Seq2Seq (Herzig and Berant, 2021)	99.9	11.6	0
+ ELMo (Herzig and Berant, 2021)	**100**	54.9	41.6
BERT2Seq (Herzig and Berant, 2021)	**100**	77.7	95.3
GRAMMAR (Herzig and Berant, 2021)	**100**	0.0	4.2
BART (Herzig and Berant, 2021)	**100**	50.5	**100**
SpanBasedSP (Herzig and Berant, 2021)	**100**	**100**	**100**
Our approach			
Denotation accuracy	**100**	**100**	**100**
↳ w/o CPLEX heuristic	**100**	**100**	**100**

Table 5.1: Denotation accuracy on the test sets for SCAN. For our approach, we also report the accuracy without the use of CPLEX to round non-integral solutions, *i.e.* they are considered as wrong predictions.

5.4.2 Baselines

To evaluate our approach, we compare it against several baselines. On all three datasets, we compare ourselves to SpanBasedSP (Herzig and Berant, 2021) as well as their sequence-to-sequence baselines. In Seq2Seq (Jia and Liang, 2016), the sentence is encoded with pre-trained GloVe (Pennington et al., 2014) or ELMo (Peters et al., 2018) embeddings followed by a bi-LSTM. The decoder is an attention-based LSTM (Bahdanau et al., 2015). BERT2Seq replaces the encoder of the Seq2Seq baseline by `BERT-base`. GRAMMAR uses the same architecture as Seq2Seq but the decoding is constrained by a grammar. Finally, BART (Lewis et al., 2020) is pre-trained as a denoising autoencoder first.

We use two additional baselines on the GeoQuery dataset. The first one is LeAR (Liu et al., 2021), a semantic parser introduced in Section 2.3.2 that relies on a Tree-LSTM (Tai et al., 2015) encoder. The second one is an approach proposed by Lindemann et al. (2023). It consists of a fertility step, predicting the number of output tokens generated by each word, followed by a reordering step. Each token is then predicted independently.

	Clevr	
	IID	Closure
Baselines (denotation accuracy)		
Seq2Seq (Herzig and Berant, 2021)	**100**	59.5
+ ELMo (Herzig and Berant, 2021)	**100**	64.2
BERT2Seq (Herzig and Berant, 2021)	**100**	56.4
GRAMMAR (Herzig and Berant, 2021)	**100**	51.3
BART (Herzig and Berant, 2021)	**100**	51.5
SpanBasedSP (Herzig and Berant, 2021)	96.7	98.8
Our approach		
Denotation accuracy	**100**	**99.6**
↳ w/o CPLEX heuristic	**100**	98.0

Table 5.2: Denotation accuracy on the test sets for Clevr. For our approach, we also report the accuracy without the use of CPLEX to round non-integral solutions, *i.e.* they are considered as wrong predictions.

5.4.3 Experimental results

SCAN. We report the denotation accuracies on the SCAN dataset in Table 5.1. We observe that both our approach and SpanBasedSP reach a perfect accuracy on all three splits. On the other hand, the sequence-to-sequence baselines suffer from a significant drop in accuracy on the splits that require compositional generalization.

Clevr. We report the denotation accuracies on the Clevr dataset in Table 5.2. Once again, while the sequence-to-sequence baselines reach a perfect accuracy on the IID split, their accuracy drops by 35.6 to 48.7 points on the Closure split. SpanBasedSP fails to reach a perfect accuracy on the IID split but generalizes very well on the Closure split. Our approach outperforms every baseline on both splits.

GeoQuery. We report the denotation accuracies for GeoQuery in Table 5.3. We observe that the approach based on reordering proposed by Lindemann et al. (2023) is competitive with the span-based approaches, *i.e.* SpanBasedSP and LeAR. Once again, our approach outperforms every baseline. However, we note that the accuracy is still significantly worse on the split that requires to generalize to longer programs. For this dataset,

	GeoQuery		
	IID	Template	Length
Baselines (denotation accuracy)			
Seq2Seq (Herzig and Berant, 2021)	78.5	46.0	24.3
+ ELMo (Herzig and Berant, 2021)	79.3	50.0	25.7
BERT2Seq (Herzig and Berant, 2021)	81.1	49.6	26.1
GRAMMAR (Herzig and Berant, 2021)	72.1	54.0	24.6
BART (Herzig and Berant, 2021)	87.1	67.0	19.3
SpanBasedSP (Herzig and Berant, 2021)	86.1	82.2	63.6
LeAR (Liu et al., 2021)	-	84.1	-
Lindemann et al. (2023)	89.1	80.4	68.8
Our approach			
Denotation accuracy	**92.9**	**89.9**	**74.9**
Exact match	90.7	86.2	69.3
↳ w/o CPLEX heuristic	90.0	83.0	67.5

Table 5.3: Denotation accuracy on the test sets for GeoQuery. For our approach, we also report the exact match accuracy and the exact match accuracy without the use of CPLEX to round non-integral solutions, *i.e.* they are considered as wrong predictions.

we also report the exact match accuracy for our approach to provide a baseline that does not require executing the semantic programs. As it is slightly lower than the denotation accuracy, this means that our approach sometime predicts a program that is different from the gold program but still returns the same denotation.

5.5 Conclusion

In this chapter, we focused on graph-based semantic parsing for formalisms that do not allow reentrancy, *i.e.* their graph representations are trees. Unlike previous work, we propose to predict the vertices and arcs of the graph jointly such that it is well-formed with respect to the semantic grammar and without restricting the search space. We proved that the problem is NP-hard and proposed an ILP formulation with a solver for its linear relaxation via the conditional gradient method.

Experimentally, our approach outperforms comparable baselines on all the datasets considered. However, even if our graph-based semantic parser provides better results

than previous work on length generalization, this setting is still difficult. Developing neural architectures that generalize better to longer sentences is thus an important general research direction for future work.

Chapter 6

Improving structural generalization via supertagging

In Chapter 5, we proposed a graph-based semantic parser that predicts the entire graph jointly. We relied on the assumption that the semantic graphs were trees. However, these assumptions do not hold in every case. Recent datasets that explicitly test for compositional generalization like COGS (Kim and Linzen, 2020) and CFQ (Keysers et al., 2020) require a parser to be able to produce graphs that potentially contain cycles or reentrancies. In addition, COGS was explicitly designed to evaluate separately lexical generalizations, *i.e.* words that appear in a role unseen during training, and structural generalizations, *i.e.* novel grammatical structures. While the majority of recent semantic parsers perform well on lexical generalizations, structural ones remain challenging.

Our novel contribution is an inference pipeline with an additional supertagging step that constrains the prediction of arc labels in the semantic graph. In Section 6.1, we motivate our approach by presenting recent graph-based approaches for COGS and CFQ with their respective strengths and weaknesses. Then, we define semantic supertagging and introduce our approach in Section 6.2. A formal definition of our novel pipeline is given in Section 6.3. Notably, we prove that our supertagging step is NP-hard and present an integer linear programming formulation for it. In Section 6.4, we propose a relaxed variant to our approach that is more flexible and can be beneficial in specific situations. We evaluate our approach on COGS and CFQ in Section 6.5 and observe that it outperforms

comparable baselines and is almost on par with baselines that require significant manual work from domain experts. Part of the contributions in this chapter were published in Petit et al. (2023).

6.1 Motivation

As we mentioned, the COGS dataset has been explicitly designed to evaluate both lexical and structural generalizations. Graph-based semantic parsers like LaGR (Jambor and Bahdanau, 2022) demonstrated better generalization abilities than sequence-to-sequence architectures on lexical generalization cases. However, they still fail on structural generalization cases. Recent work by Weißenhorn et al. (2022) has relied on the AM parser (Groschwitz et al., 2018) and predicts supertags representing nodes of the semantic graph with their adjacent arcs. This approach yielded better results on structural generalization cases compared to other approaches. Additionnally, our contributions presented in Chapter 5 as well as previous work by Herzig and Berant (2021) have shown that introducing valency constraints in a structured decoder improved compositional generalization capabilities.

This motivates us to explore a different method for compositional generalization based on supertagging in this chapter. We propose to constrain the prediction of arc labels in the semantic graph by introducing an intermediate step in the inference pipeline, which we prove to be NP-complete. As introducing a supertagging step in a parser may lead to infeasible solutions, we propose an integer linear programming formulation of supertagging that ensures the existence of at least one feasible parse in the search space, the so-called companionship principle (Bonfante et al., 2009, 2014). We also highlight that this does not impact the final step of the pipeline as it can be reduced to a matching problem.

6.2 Supertagging for graph-based semantic parsing

In this section, we first introduce the notion of supertagging. Then, we present a way to construct a set of supertags for graph-based semantic parsing. Finally, we propose a novel inference pipeline that relies on supertag prediction to improve structural generalization.

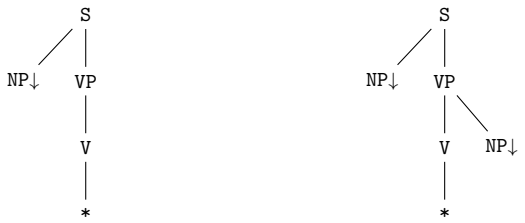

Figure 6.1: Two supertag examples from an LTAG. **(left)** Supertag associated with an intransitive verb. The substitution site NP↓ indicates the position of the subject. **(right)** Supertag associated with a transitive verb. The supplementary substitution site on the right indicates the position of the object of the verbal phrase.

6.2.1 Supertagging

In the syntactic parsing literature, supertagging refers to assigning complex descriptions of the syntactic structure directly at the lexical level (Bangalore and Joshi, 1999). For example, while an occurrence of the verb "to walk" can be described in a coarse manner via its part-of-speech tag, a supertag additionally indicates that this verb appears in a clause with a subject on the left and a verbal phrase on the right, the latter also potentially requiring an object on its right, see Figure 6.1 for an illustration in the formalism of Lexicalized Tree-Adjoining Grammars (LTAGs, Joshi et al., 1975).

6.2.2 Semantic supertagging

We propose to introduce an intermediary *semantic supertagging* step in a graph-based semantic parser.

> **Definition 6.1: Semantic supertag**
>
> A semantic supertag indicates the expected arguments of a concept (potentially none for an entity) and how that concept is used. An expected argument is referred to as a substitution site and an expected usage as a root. Formally, it is defined as a multiset of tuples $(l, d) \in L \times \{-, +\}$ where l is a label and d indicates either a substitution site (with $-$) or a root (with $+$).

In this chapter, we denote the set of all supertags as S. Note that contrary to syntactic grammars, our supertags do not impose a direction or an ordering on their labels.

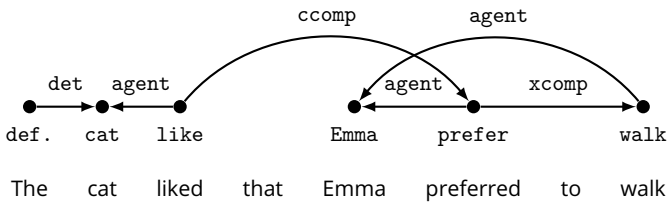

Figure 6.2: The semantic graph corresponding to the sentence "The cat liked that Emma preferred to walk" in the COGS dataset. Examples of semantic supertags in this graph are given in Example 6.2.

Example 6.2: Semantic supertags

The graph corresponding to "The cat liked that Emma preferred to walk" in the COGS dataset is shown in Figure 6.2. For example, the supertag associated with the concept `like` is $[\![(\texttt{agent}, -), (\texttt{ccomp}, -)]\!]$ and the one associated with the concept `prefer` is $[\![(\texttt{agent}, -), (\texttt{xcomp}, -), (\texttt{ccomp}, +)]\!]$.

Definition 6.3: Companionship principle

The companionship principle states that each substitution site must have a potential root in the supertag sequence, *i.e.* the number of substitution sites with a given label exactly matches the number of roots with that label.

We borrowed the name "companionship principle" from Bonfante et al. (2009, 2014) although our usage is slightly different.

Example 6.4: Failure to satisfy the companionship principle

Let us assume that we would like to parse the sentence "Marie ate". If we associate the transitive supertag $[\![(\texttt{agent}, -), (\texttt{theme}, -)]\!]$ to the verb, parsing will fail as it doesn't have an object in that sentence.

Supertag extraction. To improve generalization capabilities, we define the set of supertags as containing the set of all observed supertags in the training set, augmented with the cross-product of all root combinations and substitution site combinations. For example, if the supertags $[\![(\texttt{ccomp}, +), (\texttt{agent}, -)]\!]$ and $[\![(\texttt{agent}, -), (\texttt{theme}, -)]\!]$ are contained in the training data, we also include $[\![(\texttt{ccomp}, +), (\texttt{agent}, -), (\texttt{theme}, -)]\!]$ and $[\![(\texttt{agent}, -)]\!]$ in

the set of supertags.

Formally, let S^+ be the set of root combinations and S^- the set of substitution site combinations observed in the data. The set of supertags is:

$$S = \left\{ s^+ \cup s^- \;\middle|\; \begin{array}{l} s^+ \in S^+ \;\wedge\; s^- \in S^- \\ \wedge\; s^+ \cup s^- \neq [\![]\!] \end{array} \right\}$$

Note that the empty set can not be a supertag, as this could prevent a concept instance from appearing in the semantic structure.

6.2.3 Inference pipeline

With our novel supertagging step, the inference pipeline becomes:

1. Concept tagging
2. Semantic supertagging
3. Argument identification

We illustrate it in Figure 6.3. Note that the concept tagging step remains unchanged with respect to the traditional pipeline presented in Chapter 3. Let $w = (w_1, \ldots, w_n)$ be a sentence, $G = \langle V, A \rangle$ a complete graph constructed over w and E be the set of concepts, including a special tag $\emptyset \in E$ that is used to identify semantically empty words, *i.e.* words that do not trigger any concept. Let $\boldsymbol{\mu} \in \mathbb{R}^{|V| \times |E|}$ be tag weights computed by the neural network. We denote a sequence of tags as a boolean vector $\boldsymbol{x} \in \{0,1\}^{|V| \times |E|}$ where $x_{v,e} = 1, e \neq \emptyset$ indicates that the vertex v is tagged with the concept $e \in E$. This means that $\forall v \in V, \sum_{e \in E} x_{v,e} = 1$. Given weights $\boldsymbol{\mu}$, computing the sequence of tags of maximum linear weight is a simple problem. The two remaining steps are presented in the following section.

6.3 Mathematical formulation and resolution

In the previous section, we defined semantic supertagging and proposed to introduce it as an intermediate step in our inference pipeline. In this section, we first prove that this

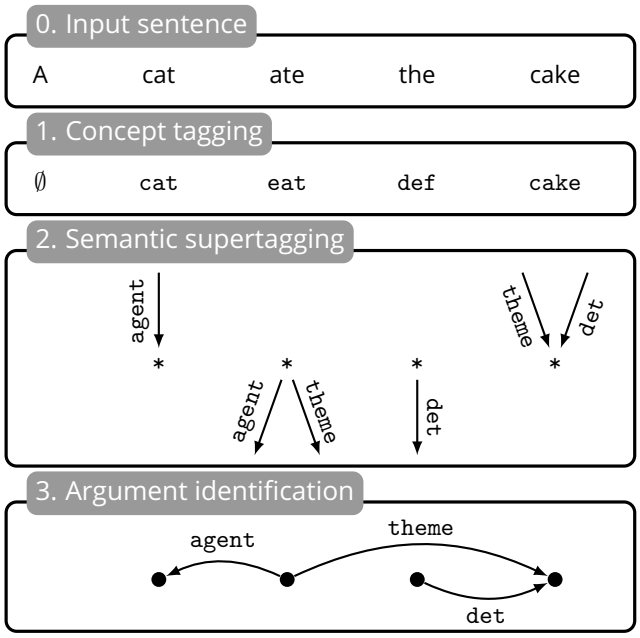

Figure 6.3: Illustration of our novel inference pipeline on the sentence "A cat ate the cake". We first predict at most one concept per word. The second step assigns a supertag to each word tagged with a concept that is not $\emptyset$. Finally, arguments are identified using the valency constraints imposed by the supertags.

problem is NP-complete. Then, we present an integer linear programming (ILP) formulation for it. Finally, we explain how this supertagging step impacts the final step of the inference pipeline.

6.3.1 NP-completeness of supertagging with companionship principle

> **Theorem 6.5: NP-completeness of supertagging with companionship principle**
>
> The following problem is NP-complete: is there a sequence of supertag assignments with linear weight $\geq m$ that satisfies the companionship principle?

Proof. First, note that given a sequence of supertags, it is trivial to check in linear time that its linear weight is $\geq m$ and that it satisfies the companionship principle, therefore the problem is in NP. We now prove NP-completeness by reducing 3-dimensional matching to supertagging with the companionship principle.

3-dimensional matching is defined as follows: Let $A = \{a(i)\}_{i=1}^{n}$, $B = \{b(i)\}_{i=1}^{n}$ and $C = \{c(i)\}_{i=1}^{n}$ be 3 sets of n elements and $D \subseteq A \times B \times C$. A subset $D' \subseteq D$ is a 3-dimensional matching if and only if, for any two distinct triples $(a, b, c) \in D'$ and

$(a', b', c') \in D'$, the following three conditions hold: $a \neq a'$, $b \neq b'$ and $c \neq c'$.

The following decision problem is known to be NP-complete (Karp, 1972): given A, B, C and D, is there a 3-dimensional matching $D' \subseteq D$ with $|D'| \geq n$?

We reduce this problem to supertagging with the companionship principle as follows. We construct an instance of the problem such that there are $3n$ concept instances $a(1), ..., a(n), b(1), ..., b(n), c(1), ..., c(n)$. The set of supertags S is defined as follows, where the weight associated to a supertag is 0 except if stated otherwise:

- For each triple $(a, b, c) \in D$, we add a supertag $[\![(b, -), (c, -)]\!]$ to S with weight 1 if and only if it is predicted for concept a;
- For each $b \in B$, we add a supertag $[\![(b, +)]\!]$ to S with weight 1 if and only if it is predicted for concept b;
- For each $c \in C$, we add a supertag $[\![(c, +)]\!]$ to S with weight 1 if and only if it is predicted for concept c.

If there exists a sequence of supertag assignment satisfying the CP that has a weight $\geq m = 3n$, then there exists a solution for the 3-dimensional matching problem, given by the supertags associated with concept instances $a(1), ..., a(n)$.

□

Theorem 6.5 motivates the use of an heuristic based method to predict supertags. We rely on the linear relaxation of an integer linear programming formulation of the problem, that we embed in a branch-and-bound procedure to retrieve the optimal solution.

6.3.2 Supertagging mathematical program

Let $\boldsymbol{z}^- \in \{0, 1\}^{|V| \times |S^-|}$ and $\boldsymbol{z}^+ \in \{0, 1\}^{|V| \times |S^+|}$ be indicator variables of the substitution sites and roots, respectively, associated with each word, *e.g.* $z^-_{v,s} = 1$ indicates that the vertex $v \in V$ has substitution sites $s \in S^-$. Let $\boldsymbol{\lambda}^- \in \mathbb{R}^{|V| \times |S^-|}$ and $\boldsymbol{\lambda}^+ \in \mathbb{R}^{|V| \times |S^+|}$ be supertag weights computed by the neural network. We now describe the constraints that $\boldsymbol{z}^-$ and $\boldsymbol{z}^+$ must satisfy. First, each vertex should have exactly one set of substitution sites

and one set of roots if and only if they have an associated concept that is not $\emptyset$:

$$\sum_{s \in S^-} z^-_{v,s} = 1 - x_{v,\emptyset} \qquad \forall v \in V \tag{6.1}$$

$$\sum_{s \in S^+} z^+_{v,s} = 1 - x_{v,\emptyset} \qquad \forall v \in V \tag{6.2}$$

Next, we forbid the empty supertag:

$$z^-_{v,[\![]\!]} + z^+_{v,[\![]\!]} \leq 1 \qquad \forall v \in V \tag{6.3}$$

Finally, we need to enforce the companionship principle. We count in $v^-_{s,l}$ the number of substitution sites with label $l \in L$ in $s \in S^-$, and similarly in $v^+_{s,l}$ for roots. We can then enforce the number of roots with a given label to be equal to the number of substitution sites with the same label as follows:

$$\sum_{\substack{v \in V, \\ s \in S^-}} z^-_{v,s} v^-_{s,l} = \sum_{\substack{v \in V, \\ s \in S^+}} z^+_{v,s} v^+_{s,l} \qquad \forall l \in L \tag{6.4}$$

All in all, supertagging with the companionship principle reduces to the following integer linear program:

$$\begin{aligned} \max_{\boldsymbol{z}^-, \boldsymbol{z}^+} \quad & \langle \boldsymbol{z}^-, \boldsymbol{\lambda}^- \rangle + \langle \boldsymbol{z}^+, \boldsymbol{\lambda}^+ \rangle, \\ \text{s.t.} \quad & (6.1\text{–}6.4), \\ & \boldsymbol{z}^- \in \{0,1\}^{V \times S^-}, \boldsymbol{z}^+ \in \{0,1\}^{V \times S^+}. \end{aligned}$$

In practice, we use the CPLEX solver.[1]

6.3.3 Argument identification

The last step of the pipeline is argument identification. Note that in many cases, there is no ambiguity, see the example in Figure 6.3: as there is at most one root and substitution site per label, we can infer that the theme of `eat` is `cake`, etc. However, in the general case, there may be several roots and substitution sites with the same label. In the example of

[1] https://www.ibm.com/products/ilog-cplex-optimization-studio

Figure 6.2, we would have 3 `agent` roots after the supertagging step.

For ambiguous labels after the supertagging step, we can rely on a bipartite matching (or assignment) algorithm. Let $\boldsymbol{\phi} \in \mathbb{R}^{|A| \times |L|}$ be the label weights computed by the neural network and $l \in L$ be an ambiguous label. We construct a bipartite undirected graph as follows:

- The first node set C contains one node per substitution site with label l
- The second node set C' contains one node per root with label l
- We add an edge for each pair $(c, c') \in C \times C'$ with weight $\phi_{c \to c', l}$

We then use the Jonker-Volgenant algorithm (Jonker and Volgenant, 1988; Crouse, 2016) to compute the matching of maximum linear weight with complexity cubic with respect to the number of nodes. Note that thanks to the companionship principle, there is always at least one feasible solution to this problem, *i.e.*, our approach will never lead to a "dead-end" and will always predict a (potentially wrong) semantic parse for any given input.

6.4 Valency-relaxed pipeline

In the previous sections, we proposed a pipeline for semantic parsing with a novel supertagging step. However, as the supertags are extracted from the training set, a model cannot predict unseen root combinations or substitution site combinations at test time. Thus, we present in this section a variant of our approach that does not constrain the valency of each label in a supertag.

6.4.1 Valency-relaxed supertags

In Definition 6.1, a supertag was defined as a multiset of tuples $(l, d) \in L \times \{-, +\}$. Intuitively, if a label l was expected for multiple roots or substitution sites, the supertag would contain multiple identical tuples $(l, +)$ or $(l, -)$ respectively. We define a valency-relaxed supertag as a semantic supertag such that it contains a tuple $(l, +)$ (respectively $(l, -)$) if and only if l is the label of at least one root (respectively substitution site). As such, for any two tuples (l_i, d_i) and (l_j, d_j) in a valency-relaxed supertag s, we have either $l_i \neq l_j$

or $d_i \neq d_j$. Similarly to Section 6.2, we define the set of valency-relaxed supertags S as the set of all valency-relaxed supertags observed in the training set augmented with the product of all root combinations and substitution site combinations.

> **Example 6.6: Valency-relaxed supertag**
>
> We depict in Figure 6.2 the semantic graph associated to the sentence "The cat liked that Emma preferred to walk" in the COGS dataset. The semantic supertag associated to the concept `Emma` is $[\![(\texttt{agent}, +), (\texttt{agent}, +)]\!]$. Its valency-relaxed supertag is $[\![(\texttt{agent}, +)]\!]$ which indicates that "`agent`" is the label corresponding to at least one root but does not specify how many.

6.4.2 Valency-relaxed supertag prediction

Let $\boldsymbol{z}^- \in \{0,1\}^{|V|\times|S^-|}$ and $\boldsymbol{z}^+ \in \{0,1\}^{|V|\times|S^+|}$ be indicator variables of the substitution sites and roots, respectively, associated with each word, *i.e.* $z^-_{v,s} = 1$ indicates that the vertex $v \in V$ has at least one of each substitution site in $s \in S^-$. We now describe the new constraints that $\boldsymbol{z}^-$ and $\boldsymbol{z}^+$ must satisfy in this variant. Each position in the sentence still requires exactly one set of substitution sites and one set of roots if and only if they have an associated concept:

$$\sum_{s\in S^-} z^-_{v,s} = 1 - x_{v,[\![]\!]} \qquad \forall v \in V \tag{6.5}$$

$$\sum_{s\in S^+} z^+_{v,s} = 1 - x_{v,[\![]\!]} \qquad \forall v \in V \tag{6.6}$$

The empty supertag is still forbidden:

$$z^-_{v,[\![]\!]} + z^+_{v,[\![]\!]} \leq 1 \qquad \forall v \in V \tag{6.7}$$

Finally, we need to ensure that the set of supertags predicted do not yield an infeasible solution. As the supertags do not include the valency, we simply ensure that if there is at least one substitution site (respectively one root) with label $l \in L$, there is at least one root (respectively one substitution site) as well. We count in $v^-_{s,l}$ whether there is a substitution site with label $l \in L$ in $s \in S^-$, and similarly in $v^+_{s,l}$ for roots. We can enforce our constraint as follows:

$$\sum_{\substack{v \in V,\\ s \in S^-}} z^-_{v,s} v^-_{s,l} \leq n \sum_{\substack{v \in V,\\ s \in S^+}} z^+_{v,s} v^+_{s,l} \qquad \forall l \in L \tag{6.8}$$

$$\sum_{\substack{v \in V,\\ s \in S^+}} z^+_{v,s} v^+_{s,l} \leq n \sum_{\substack{v \in V,\\ s \in S^-}} z^-_{v,s} v^-_{s,l} \qquad \forall l \in L \tag{6.9}$$

All in all, valency-relaxed semantic supertagging reduces to the following integer linear program:

$$\begin{aligned} \max_{\boldsymbol{z}^-, \boldsymbol{z}^+} \quad & \langle \boldsymbol{z}^-, \boldsymbol{\lambda}^- \rangle + \langle \boldsymbol{z}^+, \boldsymbol{\lambda}^+ \rangle, \\ \text{s.t.} \quad & \text{(6.5–6.9)}, \\ & \boldsymbol{z}^- \in \{0,1\}^{V \times S^-}, \boldsymbol{z}^+ \in \{0,1\}^{V \times S^+} \end{aligned}$$

6.4.3 Valency-relaxed argument identification

Let $\boldsymbol{y} \in \{0,1\}^{|A| \times |L|}$ be an indicator variable of labeled arcs, *i.e.* $y_{u \to v, l} = 1$ indicates that there is an arc labeled with l from the vertex u to the vertex v, and $\boldsymbol{\phi} \in \mathbb{R}^{|A| \times |L|}$ be label weights computed by the neural network.

As we do not know the valencies of the supertags, we aim to predict at least one arc

labeled l for each supertag in which it appears:

$$\sum_{u \in V} y_{v \to u,l} \geq \sum_{\substack{u \in V, \\ s \in S^-}} z^-_{u,s} v^-_{s,l} \qquad \forall v \in V, l \in L \tag{6.10}$$

$$\sum_{u \in V} y_{v \to u,l} \leq n \sum_{\substack{u \in V, \\ s \in S^-}} z^-_{u,s} v^-_{s,l} \qquad \forall v \in V, l \in L \tag{6.11}$$

$$\sum_{u \in V} y_{u \to v,l} \geq \sum_{\substack{u \in V, \\ s \in S^+}} z^+_{u,s} v^+_{s,l} \qquad \forall v \in V, l \in L \tag{6.12}$$

$$\sum_{u \in V} y_{u \to v,l} \leq n \sum_{\substack{u \in V, \\ s \in S^+}} z^+_{u,s} v^-_{s,l} \qquad \forall v \in V, l \in L \tag{6.13}$$

These constraints ensure that if a label l appears in a supertag, it will result in at least one arc and at most n arcs while arcs cannot be created if it is not present. Our problem is thus reduced to solving the following ILP:

$$\begin{aligned} \max_{\boldsymbol{y}} \quad & \langle \boldsymbol{y}, \boldsymbol{\phi} \rangle \\ \text{s.t.} \quad & (6.10\text{–}6.13) \end{aligned}$$

Note that the solution to this ILP can contain parallel arcs. Indeed, if two labels both have a single root on the same position in the sentence and a single substitution site on the same position in the sentence, satisfying these supertags requires parallel arcs. In that case, we only keep a single arc with the label l of maximum weight.

Intuitively, this step is almost identical to the traditional argument identification step but it can enforce the prediction of some additional arcs even though they have a negative weight.

6.5 Experiments

6.5.1 Experimental setup

Neural architecture. In our experiments, a neural architecture was used to obtain the weights $\boldsymbol{\mu}$, $\boldsymbol{\phi}$, $\boldsymbol{\lambda}^+$ and $\boldsymbol{\lambda}^-$. Our baseline architecture is as follows:

- A contextual representation for each word is obtained with an embedding layer of dimension 200 followed by a bi-LSTM with a hidden size of 400.
- This representation is given to a linear projection of dimension 300 followed by a ReLU activation and another projection of dimension $|E|$ to obtain $\boldsymbol{\mu}$.
- It is given to a linear projection of dimension 200 followed by a ReLU activation and another projection of dimension $|S^+|$ to obtain $\boldsymbol{\lambda}^+$.
- It is given to another linear projection of dimension 200 followed by a ReLU activation and a projection of dimension $|S^-|$ to obtain $\boldsymbol{\lambda}^-$.
- Finally, the representation is given to a linear projection of dimension 200 followed by a ReLU activation and a bi-affine layer (Dozat and Manning, 2017) to obtain $\boldsymbol{\phi}$.

We apply dropout with a probability of 0.3 on the output of the bi-LSTM and after the ReLU activations. The learning rate is 5×10^{-4} and each batch is composed of 30 sentences. For experiments in which we do not use the early stopping strategy presented below, we keep the parameters that obtain the best accuracy on the development set after 20 epochs.

Training objective. In Chapter 4, we presented a weakly-supervised loss for graph-based semantic parsing. Notably, we relied on an upper bound to the log-partition function: a sum of LSE functions over the set of concepts E for each word and the set of labels L for each pair of words. In the approach proposed in this chapter, each word is assigned exactly one root combination and one substitution site combination. Thus, we can apply the same decomposition for supertags and the upper bound to the log-partition function is:

$$\sum_v \left(\log \sum_e \exp(\mu_{v,e} x_{v,e}) \right) + \sum_a \left(\log \sum_l \exp(\phi_{a,l} y_{a,l}) \right)$$
$$+ \sum_v \left(\log \sum_s \exp(\lambda^-_{v,s} z^-_{v,s}) \right) + \sum_v \left(\log \sum_s \exp(\lambda^+_{v,s} z^+_{v,s}) \right)$$

To compute the anchoring of maximum weight, we follow the approach based on factor graphs presented in Section 4.3, *i.e.* the supertag weights $\boldsymbol{\lambda}^+$ and $\boldsymbol{\lambda}^-$ are not used for this computation.

Early stopping. COGS only possesses an in-distribution development set and the accuracy of most parsers on this set usually reaches 100%. Previous work by Conklin et al. (2021) highlighted that the lack of a development set representative of the generalization set makes model selection difficult and hard to reproduce. They proposed to sample a small subset of the generalization set that is used for development. Both their work and LaGR (Jambor and Bahdanau, 2022) use this approach and sample 1000 sentences from the generalization set to use as their development set. However, we argue that this set leaks compositional generalization information during training.

We propose a variant of early stopping to prevent overfitting on the in-domain data without requiring a compositional generalization development set. In this variant, we incrementally freeze the layers of the neural network as follows: each subtask (prediction of concepts, supertags, dependencies) is monitored independently on the in-domain development set. As soon as one of these tasks achieves 100% accuracy, we freeze the shared part of the neural architecture, *i.e.* the embedding layer and the bi-LSTM. We also freeze the layers that produce the scores of the perfectly predicted task. For each subsequent task that achieves perfect accuracy, the corresponding layers are also frozen. This early stopping approach aims to prevent overfitting.

6.5.2 Baselines

To assess the impact of our method, we compare it to several baselines, both existing work as well as simpler versions of our model.

Sequence-to-sequence baselines. For the COGS dataset, Kim and Linzen (2020) evaluated a LSTM-based and a Transformer-based model to serve as a baseline. An approach based on data augmentation was proposed by Akyürek et al. (2021). Zheng and Lapata (2021) proposed to introduce a semantic tagging step in their model. The encoder then uses a concatenation of the embeddings of each word and its corresponding semantic tag to encoder the sentence.

For the CFQ dataset, we report the Transformer-based models proposed by Keysers et al. (2020) and Furrer et al. (2020). We also report the approach proposed by Herzig et al. (2021) which relies on a T5 model and intermediate representations. For this baseline, we

report the accuracy when the T5 model is not pretrained as this could leak information and lead to an overestimation of the model's ability to generalize (Kim et al., 2022).

Compositional baselines. For COGS, we use the two graph-based baselines presented in Section 3.4: LaGR (Jambor and Bahdanau, 2022), which predicts every concept and label independently, and the AM parser (Weißenhorn et al., 2022), which predicts a supertag representing a tagged vertex and its adjacent labeled arcs for each word. We also evaluate our approach against LeAR (Liu et al., 2021), a semantic parser introduced in Section 2.3.2 that relies on a Tree-LSTM (Tai et al., 2015) encoder. We also report the accuracy of this parser when the encoder is replaced by a bi-LSTM.

For the CFQ dataset, we report the accuracies reached by LaGR and LeAR. Our final baseline is the approach proposed by Guo et al. (2020) based on Hierarchical Poset Decoding (HPD). Their multi-step decoding process relies on a lexicon for the prediction of concepts. As such, we also report the accuracy of their method when the hierarchical mechanism is ablated.

Our baselines. Our approach introduces a novel supertagging step in the inference pipeline as well as an early stopping strategy to prevent overfitting. We aim to properly assess their impact. Regarding our supertagging step, we report the accuracy of a standard graph-based parser as well as the accuracy of a parser that uses supertagging as an auxiliary loss but not during inference. Candito (2022) highlighted that auxiliary loss functions improved the performance of models for dependency parsing. We also report the accuracy of our baselines without early stopping.

6.5.3 Experimental results on COGS

We report the exact match accuracies on the COGS dataset in Table 6.1. We report the overall accuracy, the accuracy over all lexical generalizations as well as the accuracy over each structural generalization. Note that the overall accuracy mostly reflects the accuracy on lexical generalizations as they represent 85.7% of the test set. We observe that our approach outperforms every baseline except LeAR which relies on hand-crafted semantic operations. Importantly, our method achieves high exact match accuracy on the

	Structural			Lexical	Overall
	Obj to Subj PP	PP rec.	CP rec.		
Sequence-to-sequence models					
Kim and Linzen (2020)	0	0	0	42	35
Conklin et al. (2021)†	-	-	-	-	67
Akyürek et al. (2021)	0	1	0	96	83
Zheng and Lapata (2021)	0	39	12	99	89
Structured models					
LeAR (Liu et al., 2021)	-	-	-	-	97.7
w/o Tree-LSTM	-	-	-	-	80.7
reprod. by Weißenhorn et al. (2022)	**93**	99	**100**	99	**99**
Jambor and Bahdanau (2022)†	-	-	-	-	82.3
Weißenhorn et al. (2022)	59	36	**100**	82	79.6
Our baselines: Standard graph-based parser					
Full model	11.6	0	0	97.4	84.1
w/o early stopping	12.7	0	0	97.3	84.1
w/o early stop. & w/o supertagging loss	9.8	0	0	97.5	84.1
Proposed method: graph-based parser with supertagging					
Full model	75.0	**100**	**100**	**99.1**	98.1
w/o early stopping	51.1	**100**	**100**	98.9	96.7

Table 6.1: Exact match accuray on COGS. We report results for each subset of the test set (structural generalization and lexical generalization) and the overall accuracy. For our results, we report the mean over 3 runs. Entries marked with † use a subset of 1k sentences from the generalization set as their development set.

structural generalization examples, although the *Obj to Subj PP* generalization remains difficult. For this case, our approach only reaches an accuracy of 75.0%. Errors are discussed below.

Impact of semantic supertagging. We now consider the effect of our novel inference procedure compared to our standard graph-based pipeline. It predicts generalizations *PP recursion* and *CP recursion* perfectly, where the baseline accuracy for these cases is 0. For *Obj to Subj PP* generalization, our best configuration reaches an accuracy of 75.0%, 6 times more than our baselines. All in all, the proposed inference strategy improves results in the three structural generalizations subsets, and brings lexical generalization cases closer to 100% accuracy.

We report in Table 6.2 the supertagging accuracy with and without enforcing the companionship principle. We observe a sharp drop in accuracy for the *Obj to Subj PP* generalization when the companionship principle is not enforced. This highlights the importance

	Obj to Subj PP	PP rec.	CP rec.
Word level accuracy			
ILP	90.2	100	100
No ILP	71.6	99.9	100
Sentence level accuracy			
ILP	75.0	100	100
No ILP	9.0	99.6	100

Table 6.2: Supertagging accuracy using our integer linear program (ILP) and without (*i.e.* simply predicting the best supertag for each word, without enforcing the companionship principle).

of structural constraints to improve compositional generalization. We depict a sentence for which our model is mistaken in Figure 6.4. We observe that the error comes from the presence of the prepositional phrase after the subject. The supertagger wrongly assigns a `theme` root to the subject instead of an `agent` one. When the companionship principle is enforced, this mistake is corrected.

In addition, we also observe that the sentence level accuracy for the *Obj to Subj PP* generalization is 75.0%. This means that errors in our pipeline happen during the semantic supertagging step. However, if supertags are correctly predicted, argument identification is always correct.

Impact of training procedure. The early stopping approach introduced above has a clear impact for *Obj to Subj PP*, resulting in a 23.9 points increase (from 51.1 to 75.0). Such improvements are not observed for the baselines. From this, we conclude that our neural architecture tends to overfit the COGS training set and that some measures must be taken to mitigate this behaviour.

6.5.4 Experimental results on CFQ

We report the exact match accuracies on the CFQ dataset in Table 6.3. We observe that our approach significantly outperforms the sequence-to-sequence baselines as well as LaGR (Jambor and Bahdanau, 2022). HPD (Guo et al., 2020) is competitive with our model and LeAR (Liu et al., 2021) outperforms it by 25 points on average. However, one must note that HPD relies on a lexicon to guide the training of its hierarchical mechanism. LeAR

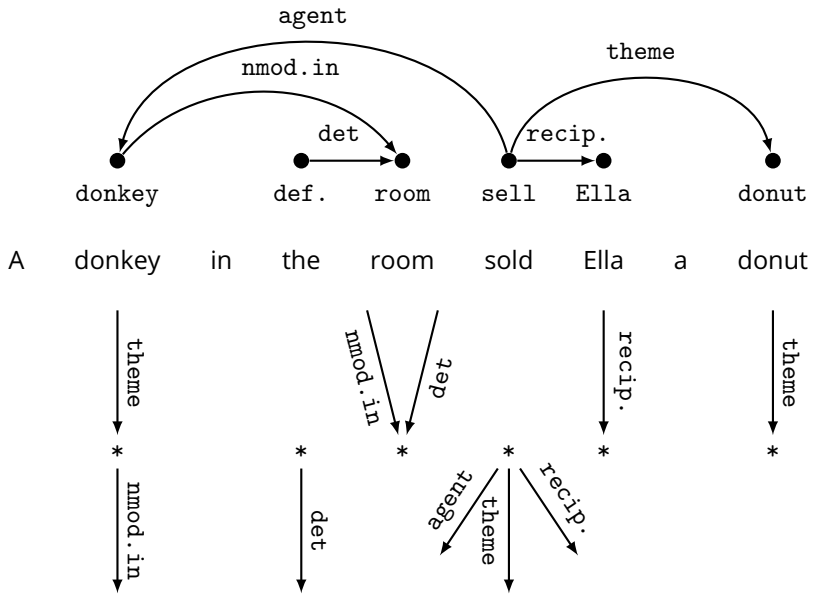

Figure 6.4: **(top)** Semantic graph of the sentence "A donkey in the room sold Ella a donut". **(bottom)** The supertags predicted by our parser if we do not enforce the companionship principle via the ILP. A mistake occurs for"donkey" as the label "`theme`" is predicted in its supertag instead of "`agent`". This is probably due to the introduction of a PP between "donkey" and "sold". This mistake is fixed when using the ILP.

relies on the same lexicon to constrain inference and on manually designed semantic operations to compute a tree structure over the sentence. In both cases, we observe a sharp drop in performance, suggesting that these parts are essential to HPD and LeAR.

Impact of semantic supertagging. We observe that our baseline without supertagging performs slightly worse than LaGR with a 36.1 mean accuracy. Even though valency-relaxed supertags do not impose as strict constraints as our original supertags, we observe that their addition significantly improve the performance of our graph-based parser as the mean accuracy increases by 29.6 points. For most models, the MCD2 and MCD3 splits are the most problematic. The introduction of valency-relaxed supertags yields the largest gains on these splits with 37.5 points and 33.8 points respectively.

6.6 Conclusion

In this chapter, we focused on improving compositional generalization (particularly structural generalization) via supertagging for graph-based semantic parsing. Unlike previous

	Mean MCD	MCD1	MCD2	MCD3
Sequence-to-sequence models				
Keysers et al. (2020)	18.9	37.4	8.1	11.3
Furrer et al. (2020)	42.1	61.6	31.3	33.3
Herzig et al. (2021)	22.6	-	-	-
Structured models				
LaGR (Jambor and Bahdanau, 2022)	39.5	62.8	30.3	25.4
HPD (Guo et al., 2020)	69.0	79.6	59.6	67.8
Without hierachical decoding	12.6	21.3	6.4	10.1
LeAR (Liu et al., 2021)	90.9	91.7	89.2	91.7
Without Tree-LSTM	30.4	40.1	25.6	25.4
Our baseline: Standard graph-based parser				
Baseline model	36.1	64.2	23.8	20.3
Proposed method: graph-based parser with supertagging				
Valency-relaxed supertagging	65.7	81.7	61.3	54.1

Table 6.3: Accuracy on the CFQ dataset. We report the accuracy over each MCD split as well as the mean of these accuracies.

work on graph-based semantic parsing (Lyu and Titov, 2018; Jambor and Bahdanau, 2022), we do not assume that every part of the semantic graph can be predicted independently. With our supertagging approach, the prediction of arc labels is constrained and done jointly. We proved that supertagging with the companionship principle is NP-complete and proposed an ILP formulation for it. Experimentally, our method significantly improves results when compositional generalization is required. It outperforms every baseline with the exception of LeAR (Liu et al., 2021) and HPD (Guo et al., 2020) that rely on a lexicon and require manual work beforehand.

Future research based on this work could focus on two main directions: first, we could focus on how to better adapt our approach to datasets like CFQ that introduce novel combinations of substitutions sites and roots in the test set. While our approach brought significant improvements compared to a graph-based parser without supertagging, it still struggles on the MCD2 and MCD3 splits. The second direction would be to extend our method for representations that require words to produce multiple concepts like the Abstract Meaning Representation (Banarescu et al., 2013). To a lesser extent, this also applies to CFQ as our proposed transformation requires merging some concepts together to be able to anchor a semantic graph on its sentence.

Chapter 7

Conclusion

7.1 Conclusion

In this thesis, we aimed to improve compositional generalization in semantic parsing via graph-based approaches. There were two major motivations to use them: first, they rely on local predictions, which have been shown to be beneficial for compositional generalization. Then, relying on graphs is attractive as it is easier to make the conversion between a semantic representation and a graph than develop a tailored approach for each semantic formalism. We first dealt with the issue of training these parsers. In our setting, the weights of concepts and labels are predicted independently from each other. We proposed a theoretically motivated approach to approximate and compute the training objective in the weakly-supervised case. Then, we presented two novel approaches to improve compositional generalization. The first one (Chapter 5) predicts the entire semantic graph jointly to find the optimal solution while taking into account constraints imposed by the semantic grammar. The second one (Chapter 6) removes the assumption that arcs can be predicted independently and introduces a semantic supertagging step in the inference pipeline to constrain arc predictions. In both cases, our goal is to ensure the well-formedness of the global prediction by introducing additional local constraints in our inference algorithms.

Combinatorial optimization and training. In Chapter 4, we presented the issues that arise when training a graph-based parser. Notably, the the anchoring of a semantic graph

on its sentence is usually not available in the training data. We proposed a training objective for this scenario that we referred to as "weakly-supervised". As the objective is intractable, we derived an approximation which requires finding the optimal anchoring of the semantic graph on the sentence. We proved this problem to be NP-hard. To tackle this, we presented two approaches. The first one is an optimization framework relying on the conditional gradient method (Frank and Wolfe, 1956) and constraint relaxation. This framework allows us to solve the linear relaxation of our problems and requires only an efficient algorithm known as the LMO that computes the optimal solution when the problematic constraints are relaxed. We proposed an LMO for trees and an adaptation that can be applied in the general case. The second one frames the anchoring problem using the formalism of factor graphs on which we perform MAP inference.

Graph-based reentrancy-free semantic parsing. In Chapter 5, we studied semantic parsing in the case where the semantic graphs are trees and the grammar of the semantic representation is known beforehand. We proposed to represent the search domain as a clustered labeled graph such that each valid semantic graph that can be produced is a generalized not-necessarily spanning arborescence of that graph. This allowed us to reduce semantic parsing in that case to computing the maximum generalized valency-constrained not-necessarily spanning arborescence. We proved that this problem is NP-hard and proposed to solve instead its linear relaxation via the conditional gradient method (Frank and Wolfe, 1956). In this case, we use the maximum spanning arborescence algorithm (Myung et al., 1995) on the contracted graph as an LMO. Experimentally, we outperformed every baseline on the three datasets considered: GeoQuery, SCAN and Clevr. This demonstrated the efficiency of our method over several domains without requiring any adaptation.

Improving structural generalization via supertagging. In Chapter 6, we studied a more general case in which the semantic graphs are not necessarily trees and we do not have access to a semantic grammar. The common inference pipeline in that case is to predict the vertices first and then predict the arcs between them. We proposed to extract semantic supertags from the training data that represent the expected arc labels

for a given vertex. At inference, we introduce a supertagging step that must satisfy the companionship principle, *i.e.* there exists at least one feasible semantic graph that satisfy the constraints imposed by the supertags. We proved that this problem is NP-complete and rely on a linear program solver for this step. Experimentally, we demonstrate that introducing a supertagging step significantly improves structural generalization as it solves perfectly two out of the three cases on the COGS dataset. Introducing additional steps in a pipeline could hurt the performance in some cases as there is an increased risk of error propagation. However, we observe that in practice, it is not the case for our approach on the datasets considered.

7.2 Future research directions

The work presented in this thesis enabled significant improvements in compositional generalization for semantic parsing. It also highlighted several promising research directions that we discuss below.

Multiple vertices per word. Our contributions are based on the assumption that each word in a sentence can produce at most one vertex in the semantic graph. While this assumption holds for the datasets considered in this thesis, it is not the case of other datasets like the AMR dataset (Banarescu et al., 2013). The development of a parser that bypasses this constraint is an important future step.

Generalization to longer sentences. We also observed that graph-based parsers still struggle when generalizing to longer sentences. In Section 5.4, we reported a 15 point drop in accuracy on the Length split of GeoQuery compared to other splits. Our experiments in Section 6.5 also highlighted that without supertagging, our baseline graph-based parser fails on structural generalization with deeper recursion. Thus, the development of neural architectures that generalize better to longer sentences could yield significant improvements.

More generic semantic supertags. Both of our contributions demonstrated that local constraints imposed either by a known grammar or supertags predicted by the model

improved the accuracy of the parser. However, this information must either be known beforehand (Chapter 5) or fully extractable from the training data (Chapter 6). Regarding the CFQ dataset, we had to construct the set of semantic supertags present in the test set to deduce that our original approach was not suitable and the valency-relaxed pipeline was needed. Proposing a more generic representation that can handle unknown supertags could be an interesting research direction.

Additional constraints inspired by supertagging. Finally, the introduction of our semantic supertags removed the assumption that the arcs of the graph should be predicted independently and increased the accuracy of the parser. One could wonder whether the assumption that concepts can be predicted independently is justified or not. We can illustrate this with the CFQ dataset. It contains concepts representing that the agent is the `director`, `writer`, ... of the theme. It also contains the opposite concepts where the agent has been `directed_by`, `written_by`, ... the theme. While it is obvious that these concepts are incompatible, they could be predicted together by our parser. Expanding the idea of supertags to other components of the semantic graph could prevent this kind of errors, likely at the expense of a more complex inference. It could be another direction to improve our graph-based parsers.

Although these potential research directions could be promising, it is also important to ensure that further work will not trade the current progress that were made for compositional generalization in exchange for other improvements.

Bibliography

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238, Sofia, Bulgaria. Association for Computational Linguistics.

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 77–89, Vancouver, Canada. Association for Computational Linguistics.

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. 2021. Learning to recombine and resample data for compositional generalization. In *International Conference on Learning Representations*.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 47–52, Sofia, Bulgaria. Association for Computational Linguistics.

I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. 1995. Natural language interfaces to databases – an introduction. *Natural Language Engineering*, 1(1):29–81.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine transla-

tion by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Dzmitry Bahdanau, Harm de Vries, Timothy J. O'Donnell, Shikhar Murty, Philippe Beaudoin, Yoshua Bengio, and Aaron C. Courville. 2019. CLOSURE: Assessing systematic generalization of CLEVR models. *CoRR*, abs/1912.05783.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

D. Barber. 2011. *Bayesian Reasoning and Machine Learning*, 04-2011 edition. Cambridge University Press. In press.

Amir Beck. 2017. *First-Order Methods in Optimization*. SIAM.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

C.M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer.

Mathieu Blondel, André F.T. Martins, and Vlad Niculae. 2020. Learning with Fenchel-Young losses. *Journal of Machine Learning Research*, 21(35):1–69.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint*

Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1455–1465, Jeju Island, Korea. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Guillaume Bonfante, Bruno Guillaume, and Mathieu Morey. 2009. Dependency constraints for lexical disambiguation. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 242–253, Paris, France. Association for Computational Linguistics.

Guillaume Bonfante, Bruno Guillaume, Mathieu Morey, and Guy Perrier. 2014. *Supertagging with Constraints*, chapter 12.

George Boole. 1854. *An Investigation of the Laws of Thought on which are Founded the Mathematical Theories of Logic and Probabilities*. Walton and Maberly.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

Marie Candito. 2022. Auxiliary tasks to boost biaffine semantic dependency parsing. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2422–2429, Dublin, Ireland. Association for Computational Linguistics.

Bob Carpenter. 1997. *Type-Logical Semantics*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representa-

tions using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton and Co., The Hague.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

John Cocke. 1970. *Programming Languages and Their Compilers: Preliminary Notes*. New York University.

Henry Conklin, Bailin Wang, Kenny Smith, and Ivan Titov. 2021. Meta-learning to compositionally generalize. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3322–3335, Online. Association for Computational Linguistics.

Caio Corro. 2020. Span-based discontinuous constituency parsing: a family of exact chart-based algorithms with time complexities from O(nˆ6) down to O(nˆ3). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2753–2764, Online. Association for Computational Linguistics.

Caio Corro. 2023. On the inconsistency of separable losses for structured prediction. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics.

Caio Corro, Joseph Le Roux, and Mathieu Lacroix. 2017. Efficient discontinuous phrase-structure parsing via the generalized maximum spanning arborescence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1644–1654, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas M. Cover. 1999. *Elements of Information Theory*. John Wiley & Sons.

David F Crouse. 2016. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1679–1696.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations*.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Christophe Duhamel, Luis Gouveia, Pedro Moura, and Mauricio Souza. 2008. Models and heuristics for a minimum arborescence problem. *Networks*, 51(1):34–47.

J.C Dunn and S Harshbarger. 1978. Conditional gradient algorithms with open loop step size rules. *Journal of Mathematical Analysis and Applications*, 62(2):432–444.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards: Mathematics and mathematical physics. B*, 71:233.

Jason Eisner. 1997. Bilexical grammars and a cubic-time probabilistic parser. In *Proceedings of the Fifth International Workshop on Parsing Technologies*, pages 54–65, Boston/Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland. Association for Computational Linguistics.

Marguerite Frank and Philip Wolfe. 1956. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

Gottlob Frege. 1956. The thought: A logical inquiry. *Mind*, 65(259):289–311.

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *CoRR*, abs/2007.08970.

Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman.

Carlos Gómez-Rodríguez, John Carroll, and David Weir. 2011. Dependency parsing schemata and mildly non-projective dependency parsing. *Computational Linguistics*, 37(3):541–586.

Jonas Groschwitz, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2018. AMR dependency parsing with a typed semantic algebra. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1831–1841, Melbourne, Australia. Association for Computational Linguistics.

Yinuo Guo, Zeqi Lin, Jian-Guang Lou, and Dongmei Zhang. 2020. Hierarchical poset decoding for compositional generalization in language. *CoRR*, abs/2010.07792.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–237, Baltimore, Maryland. Association for Computational Linguistics.

Gary G. Hendrix, Earl D. Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum. 1978. Developing a natural language interface to complex data. *ACM Trans. Database Syst.*, 3(2):105–147.

Jonathan Herzig and Jonathan Berant. 2021. Span-based semantic parsing for compositional generalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 908–921, Online. Association for Computational Linguistics.

Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. 2021. Unlocking compositional generalization in pre-trained models using intermediate representations. *CoRR*, abs/2104.07478.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada. Association for Computational Linguistics.

Dora Jambor and Dzmitry Bahdanau. 2022. LAGr: Label aligned graphs for better systematic generalization in semantic parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3295–3308, Dublin, Ireland. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Yichen Jiang and Mohit Bansal. 2021. Inducing transformer's compositional generalization ability via auxiliary sequence prediction tasks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6253–6265, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1988–1997.

Roy Jonker and Ton Volgenant. 1988. A shortest augmenting path algorithm for dense and sparse linear assignment problems. In *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR*, pages 622–622. Springer.

Aravind K Joshi, Leon S Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *Journal of computer and system sciences*, 10(1):136–163.

Laura Kallmeyer. 2010. *Parsing Beyond Context-Free Grammars*. Springer Science & Business Media.

Richard M. Karp. 1972. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA.

Tadao Kasami. 1965. *An Efficient Recognition and Syntax-Analysis Algorithm for Context-Free Languages*. Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.

Robert T. Kasper. 1989. A flexible interface for linking applications to Penman's sentence generator. In *Speech and Natural Language: Proceedings of a Workshop Held at Philadelphia, Pennsylvania, February 21-23, 1989*.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, AAAI'05, page 1062–1068. AAAI Press.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Najoung Kim, Tal Linzen, and Paul Smolensky. 2022. Uncontrolled lexical exposure leads to overestimation of compositional generalization in pretrained models.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA. Association for Computational Linguistics.

Marco Kuhlmann and Peter Jonsson. 2015. Parsing to noncrossing dependency graphs. *Transactions of the Association for Computational Linguistics*, 3:559–570.

Marco Kuhlmann and Giorgio Satta. 2014. A new parsing algorithm for Combinatory Categorial Grammar. *Transactions of the Association for Computational Linguistics*, 2:405–418.

Harold W. Kuhn. 1955. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556, Seattle, Washington, USA. Association for Computational Linguistics.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Simon Lacoste-Julien and Martin Jaggi. 2015. On the global linear convergence of Frank-Wolfe optimization variants. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882. PMLR.

Evgeny S. Levitin and Boris T. Polyak. 1966. Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, 6(5):1–50.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for Chinese POS tagging and dependency parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1180–1191, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33, Vancouver, Canada. Association for Computational Linguistics.

Percy Liang. 2013. Lambda Dependency-Based Compositional Semantics. *arXiv e-prints*, page arXiv:1309.4408.

Matthias Lindemann, Alexander Koller, and Ivan Titov. 2023. Compositional generalisation with structured reordering and fertility layers.

Chenyao Liu, Shengnan An, Zeqi Lin, Qian Liu, Bei Chen, Jian-Guang Lou, Lijie Wen, Nanning Zheng, and Dongmei Zhang. 2021. Learning algebraic recombination for compositional generalization. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1129–1144, Online. Association for Computational Linguistics.

Peter C. Lockemann and Frederick B. Thompson. 1969. A rapidly extensible language system (the REL language processor). In *International Conference on Computational Linguistics COLING 1969: Preprint No. 34*, Sånga Säby, Sweden.

João Loula, Marco Baroni, and Brenden Lake. 2018. Rearranging the familiar: Testing compositional generalization in recurrent networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 108–114, Brussels, Belgium. Association for Computational Linguistics.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 397–407, Melbourne, Australia. Association for Computational Linguistics.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. 2019. The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision. In *International Conference on Learning Representations*.

Abelardo Carlos Martínez Lorenzo, Marco Maru, and Roberto Navigli. 2022. Fully-Semantic Parsing and Generation: the BabelNet Meaning Representation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1727–1741, Dublin, Ireland. Association for Computational Linguistics.

André Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th*

Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 342–350, Suntec, Singapore. Association for Computational Linguistics.

André Martins, Noah Smith, Eric Xing, Pedro Aguiar, and Mário Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44, Cambridge, MA. Association for Computational Linguistics.

André F. T. Martins, Mário A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, and Eric P. Xing. 2011. An augmented lagrangian approach to constrained map inference. In *International Conference on Machine Learning*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Richard Montague. 1970. Universal grammar. *Theoria*, 36(3):373–398.

Raymond J Mooney. 2014. Semantic parsing: Past, present, and future. In *Presentation slides from the ACL Workshop on Semantic Parsing*.

Stefan Müller. 2016. *Grammatical theory: From transformational grammar to constraint-based approaches*. Language Science Press.

Young-Soo Myung, Chang-Ho Lee, and Dong-Wan Tcha. 1995. On the generalized minimum spanning tree problem. *Networks*, 26(4):231–241.

Radford M. Neal and Geoffrey E. Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer.

Yu Nesterov. 2005. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152.

Vlad Niculae, Caio F. Corro, Nikita Nangia, Tsvetomila Mihaylova, and André F. T. Martins. 2023. Discrete latent structure in neural networks.

Jorge Nocedal and Stephen J. Wright. 1999. *Numerical Optimization*. Springer.

Barbara Partee. 1984. Compositionality. *Varieties of formal semantics*.

Panupong Pasupat, Sonal Gupta, Karishma Mandyam, Rushin Shah, Mike Lewis, and Luke Zettlemoyer. 2019. Span-based hierarchical semantic parsing for task-oriented dialog. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1520–1526, Hong Kong, China. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alban Petit and Caio Corro. 2023. On graph-based reentrancy-free semantic parsing.

Alban Petit, Caio Corro, and François Yvon. 2023. Structural generalization in cogs: Supertagging is (almost) all you need.

Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2012. Dynamic programming for higher order parsing of gap-minding trees. In *Proceedings of the 2012 Joint Conference*

on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 478–488, Jeju Island, Korea. Association for Computational Linguistics.

Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2013. Finding optimal 1-endpoint-crossing trees. *Transactions of the Association for Computational Linguistics*, 1:13–24.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Pawel Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. 2022. Improving compositional generalization with latent structure and data augmentation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4341–4362, Seattle, United States. Association for Computational Linguistics.

Owen Rambow. 2010. The simple truth about dependency and phrase structure representations: An opinion piece. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 337–340, Los Angeles, California. Association for Computational Linguistics.

V. Venkata Rao and R. Sridharan. 2002. Minimum-weight rooted not-necessarily-spanning arborescence problem. *Networks*, 39(2):77–87.

Giorgio Satta. 1992. Recognition of linear context-free rewriting systems. In *30th Annual Meeting of the Association for Computational Linguistics*, pages 89–95, Newark, Delaware, USA. Association for Computational Linguistics.

Natalie Schluter. 2015. The complexity of finding the maximum spanning DAG and other restrictions for DAG parsing of natural language. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 259–268, Denver, Colorado. Association for Computational Linguistics.

Alexander Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24. Springer.

M. Steedman. 1996. *Surface Structure and Interpretation*. Linguistic Inquiry Series. MIT Press.

M. Steedman. 2000. *The Syntactic Process*. A Bradford book. MIT Press.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Machine Learning: ECML 2001*, pages 466–477, Berlin, Heidelberg. Springer Berlin Heidelberg.

Robert Endre Tarjan. 1977. Finding optimum branchings. *Networks*, 7(1):25–35.

Marjorie Templeton and John Burger. 1983. Problems in natural-language interface to DBMS with examples from EUFID. In *First Conference on Applied Natural Language Processing*, pages 3–16, Santa Monica, California, USA. Association for Computational Linguistics.

Cynthia A. Thompson and Raymond J. Mooney. 2003. Acquiring word-meaning mappings for natural language interfaces. *J. Artif. Int. Res.*, 18(1):1–44.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

K Vijay-Shanker and David J. Weir. 1993. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Martin J. Wainwright and Michael Irwin Jordan. 2008. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc.

David L. Waltz. 1978. An english language question answering system for a large relational database. *Commun. ACM*, 21(7):526–539.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342, Beijing, China. Association for Computational Linguistics.

Pia Weißenhorn, Lucia Donatelli, and Alexander Koller. 2022. Compositional generalization with a broad-coverage semantic parser. In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics*, pages 44–54, Seattle, Washington. Association for Computational Linguistics.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on Universal Dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723, Austin, Texas. Association for Computational Linguistics.

A.N. Whitehead and B. Russell. 1912. *Principia Mathematica*. Number vol. 2 in Principia Mathematica. University Press.

Terry Winograd. 1972. Understanding natural language. *Cognitive Psychology*, 3(1):1–191.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446.

W. Woods, Ronald Kaplan, and Bonnie Webber. 1972. The lunar sciences natural language information system.

Jingfeng Yang, Le Zhang, and Diyi Yang. 2022. SUBS: Subtree substitution for compositional semantic parsing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 169–174, Seattle, United States. Association for Computational Linguistics.

Yuekun Yao and Alexander Koller. 2022. Structural generalization is hard for sequence-to-sequence models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5048–5062, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Pengcheng Yin, Hao Fang, Graham Neubig, Adam Pauls, Emmanouil Antonios Platanios, Yu Su, Sam Thomson, and Jacob Andreas. 2021. Compositional generalization for neural semantic parsing via span-level supervised attention. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2810–2823, Online. Association for Computational Linguistics.

Daniel H. Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and Control*, 10(2):189–208.

Alp Yurtsever, Olivier Fercoq, Francesco Locatello, and Volkan Cevher. 2018. A conditional gradient framework for composite convex minimization with applications to semidefinite programming. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5727–5736. PMLR.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, page 1050–1055. AAAI Press.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687, Prague, Czech Republic. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, page 658–666, Arlington, Virginia, USA. AUAI Press.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 665–676, Valencia, Spain. Association for Computational Linguistics.

Hao Zheng and Mirella Lapata. 2021. Compositional generalization via semantic tagging. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1022–1032, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang Qu, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 680–689, Austin, Texas. Association for Computational Linguistics.

Wang Zhu, Peter Shaw, Tal Linzen, and Fei Sha. 2021. Learning to generalize compositionally by transferring across semantic parsing tasks. *arXiv preprint arXiv:2111.05013*.

Appendix A

GeoQuery lexicon and grammar

A.1 GeoQuery lexicon

Concept	Words
area_1	area, square
capital	capital, capitals
capital_all	capital, capitals
capital_1	capital, capitals
capital_2	capital, capitals
city	city, cities
city_all	city, cities
count	many, number
density_1	density
exclude	excluding, not
fewest	fewest, least
higher_2	higher
highest	highest, tallest
high_point_1	high
lake	lake, lakes
largest	biggest, largest
largest_one	largest, most
len	long, length
loc_1	in, with
loc_2	in, of
longer	longer

Concept	Words
longest	biggest, longest
lower_2	lower
lowest	lowest
most	most
mountain	mountain, peak
next_to_1	border, borders
next_to_2	border, borders, next
place	point, spot, elevation
place_all	point
population_1	population, people
river	river, rivers
river_all	river, rivers
shortest	shortest
size	big, size
smallest	smallest
smallest_one	smallest, least
state	state, states
state_all	state, states
sum	combined, total
traverse_1	through
traverse_2	through

Table A.1: Lexicon used in our contributions for the GeoQuery dataset. Each concept appearing in this table is given an extra weight for the corresponding words.

A.2 GeoQuery grammar

Concepts	Type	Arguments
`capital_all` / `cityid` / `city_all`	city	-
`riverid` / `river_all`	river	-
`mountain_all` / `placeid` / `place_all`	place	-
`stateid` / `state_all`	state	-
`countryid`	country	-
`loc_1`	loc	city river state place
`loc_2`	loc	city state country
`city`	city	city loc
`capital`	city	city loc place
`lake` / `mountain` / `place`	place	place loc
`river`	river	river loc
`state`	state	state loc
`high_point_1` / `low_point_1`	place	state
`high_point_2` / `low_point_2`	state	place
`longer` / `longest` / `shortest`	river	river
`higher_2` / `lower_2` / `highest` / `lowest`	place	place
`next_to_1`	state	state
`next_to_2`	state	river state
`traverse_1`	city state country	river
`traverse_2`	river	city state country

Table A.2: For each concept in the GeoQuery dataset, we indicate its type and the types of its expected arguments. Concepts that are grouped on the same line share the same type and expected arguments.

Concepts	Type	Arguments
capital_1	city	country
		state
capital_2	state	city
largest / smallest	city	city
	place	place
	state	state
most / fewest	city	city
	river	river
	place	place
	state	state
major	city	city
	river	river
	place	place
elevation_1	num	place
len	num	river
size	num	city
		state
		country
count	num	city
		river
		place
		state
area_1 / density_1 / population_1	num_city	city
	num_state	state
	num_country	country
largest_one / smallest_one	city	num_city
	state	num_state
sum	num	num
		num_city
		num_state
intersection / exclude	city	city, city
		city, loc
	river	river, river
	place	place, place
		place, loc
	state	state, state
		state, loc

Table A.3: Continuation of Table A.2.

Appendix B

SCAN lexicon and grammar

B.1 SCAN lexicon

Concept	Words
`after`	after
`and`	and
`around`	around
`jump`	jump
`left`	left
`look`	look
`opposite`	opposite

Concept	Words
`right`	right
`run`	run
`thrice`	thrice
`twice`	twice
`turn`	turn
`walk`	walk

Table B.1: Lexicon used in our contributions for the SCAN dataset. Each concept appearing in this table is given an extra weight for the corresponding words.

B.2 SCAN grammar

Concepts	Type	Arguments
`left` / `right`	direction	-
`around` / `opposite`	manner	-
`jump` / `look` / `run` / `turn` / `walk`	action	- direction manner, direction
`thrice` / `twice`	action	action
`after` / `and`	combination	action, action

Table B.2: For each concept in the SCAN dataset, we indicate its type and the types of its expected arguments. Concepts that are grouped on the same line share the same type and expected arguments.

Appendix C

Clevr lexicon and grammar

C.1 Clevr lexicon

Concept	Words
blue	blue
brown	brown
color	color
count	many, number
count_equal	equal, same
count_greater	greater, more
count_less	fewer, less
cube	block(s), cube(s)
cyan	cyan
cylinder	cylinder(s)
exist	any, there
gray	gray
green	green
intersect	and
large	big, large
material	made, material
metal	metal, metallic, shiny

Concept	Words
purple	purple
query	how, what
query_attribute_equal	same
red	red
relate_attribute_equal	same
relate_behind	behind
relate_front	front
relate_left	left
relate_right	right
rubber	rubber, matte
size	size, big
shape	shape
small	small, tiny
sphere	ball, sphere
union	or
yellow	yellow

Table C.1: Lexicon used in our contributions for the Clevr dataset. Each concept appearing in this table is given an extra weight for the corresponding words.

C.2 Clevr grammar

Concepts	Type	Arguments
blue / brown / cyan / gray	object	- object
green / purple / red / yellow	object	- object
cube / cylinder / sphere	object	- object
large / small	object	- object
metal / rubber	object	- object
scene	object	-
relate_behind / relate_front	object	object
relate_left / relate_right	object	object
intersect / union	object	object, object
color / material / shape / size	property	-
relate_attribute_equal	object	property, object
count / exist	query	object
count_equal / count_greater / count_less	query	object, object
query	query	property, object
query_attribute_equal	query	property, object, object

Table C.2: For each concept in the Clevr dataset, we indicate its type and the types of its expected arguments. Concepts that are grouped on the same line share the same type and expected arguments.

Appendix D

Closed form to the smoothed indicator function

In this appendix, we detail the computations that yield a closed form to the smoothed indicator function introduced in Section 4.2. This function was defined as

$$\delta^{**}_{S,\beta}(\boldsymbol{A}\boldsymbol{z}) = \max_{\boldsymbol{u}\in S} \boldsymbol{u}^\top(\boldsymbol{A}\boldsymbol{z}) - \sigma_S(\boldsymbol{u}) - \frac{\beta}{2}\|\boldsymbol{u}\|_2^2$$

Definition D.1: Proximal operator

Let $f : \mathcal{X} \to \mathbb{R}$ be a function. The proximal operator of f is the function $\operatorname{prox}_f : \mathcal{X} \to \mathbb{R}$ defined as follows:

$$\operatorname{prox}_f(\boldsymbol{u}) = \operatorname*{arg\,min}_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}) + \frac{1}{2}\|\boldsymbol{u}-\boldsymbol{x}\|^2$$

Theorem D.2: Extended Moreau decomposition

Let $f : \mathcal{X} \to \mathbb{R}$ be a function. If f is convex and lower semi-continuous, for any $\boldsymbol{x} \in \mathcal{X}$ and any $\gamma > 0$, we have:

$$\boldsymbol{x} = \operatorname{prox}_{\gamma f}(\boldsymbol{x}) + \gamma \operatorname{prox}_{\gamma^{-1} f^*}(\gamma^{-1}\boldsymbol{x})$$

Proof. A proof of Theorem D.2 was given by Beck (2017, Section 6.6). □

Let $\hat{\boldsymbol{u}}$ be the optimal value to the smoothed indicator function, we can express it as follows:

$$\begin{aligned}
\hat{\boldsymbol{u}} &= \arg\max_{\boldsymbol{u}\in S} \boldsymbol{u}^\top(\boldsymbol{A}\boldsymbol{z}) - \sigma_S(\boldsymbol{u}) - \frac{\beta}{2}\|\boldsymbol{u}\|_2^2 \\
&= \arg\min_{\boldsymbol{u}\in S} \sigma_S(\boldsymbol{u}) + \frac{\beta}{2}\|\boldsymbol{u}\|_2^2 - \boldsymbol{u}^\top(\boldsymbol{A}\boldsymbol{z}) \\
&= \arg\min_{\boldsymbol{u}\in S} \beta^{-1}\sigma_S(\boldsymbol{u}) + \frac{1}{2}\|\boldsymbol{u}\|_2^2 - \beta^{-1}\boldsymbol{u}^\top(\boldsymbol{A}\boldsymbol{z})
\end{aligned}$$

We introduce a term $\frac{1}{2}\|\beta^{-1}\boldsymbol{A}\boldsymbol{z}\|_2^2$ as it is constant with respect to $\boldsymbol{u}$

$$\begin{aligned}
&= \arg\min_{\boldsymbol{u}\in S} \beta^{-1}\sigma_S(\boldsymbol{u}) + \frac{1}{2}\|\boldsymbol{u}\|_2^2 - \beta^{-1}\boldsymbol{u}^\top(\boldsymbol{A}\boldsymbol{z}) + \frac{1}{2}\|\beta^{-1}\boldsymbol{A}\boldsymbol{z}\|_2^2 \\
&= \arg\min_{\boldsymbol{u}\in S} \beta^{-1}\sigma_S(\boldsymbol{u}) + \frac{1}{2}\left(\|\boldsymbol{u}\|_2^2 - 2\beta^{-1}\boldsymbol{u}^\top(\boldsymbol{A}\boldsymbol{z}) + \|\boldsymbol{A}\boldsymbol{z}\|_2^2\right) \\
&= \arg\min_{\boldsymbol{u}\in S} \beta^{-1}\sigma_S(\boldsymbol{u}) + \frac{1}{2}\|\beta^{-1}\boldsymbol{A}\boldsymbol{z} - \boldsymbol{u}\|_2^2 \\
&= \operatorname{prox}_{\beta^{-1}\delta_S^*}(\beta^{-1}\boldsymbol{z})
\end{aligned}$$

Using Theorem D.2, we rewrite this as

$$= \beta^{-1}\left((\boldsymbol{A}\boldsymbol{z}) - \operatorname{prox}_{\beta\delta_S}(\boldsymbol{A}\boldsymbol{z})\right)$$

Definition D.3: Proximal operator of an indicator function

Let S be a set and δ_S its indicator function. The proximal operator of δ_S is

$$\begin{aligned}
\operatorname{prox}_{\delta_S}(\boldsymbol{t}) &= \arg\min_{\boldsymbol{u}} \delta_S(\boldsymbol{u}) + \frac{1}{2}\|\boldsymbol{t} - \boldsymbol{u}\|_2^2 \\
&= \arg\min_{\boldsymbol{u}\in S} \frac{1}{2}\|\boldsymbol{t} - \boldsymbol{u}\|_2^2
\end{aligned}$$

This is known as the projection operator on the set S.

We can now compute a closed form to our problem. We first handle the general case where S is defined by a set of inequalities. Then, we handle a specific case where it is defined by a set of equalities.

Inequality case. As the set $S = \{\boldsymbol{u} | \boldsymbol{u} \leq \boldsymbol{b}\}$ is defined by a set of affine constraints, using Definition D.3, we can rewrite our proximal operator as the element-wise minimum between $\boldsymbol{A}\boldsymbol{z}$ and $\boldsymbol{b}$ that we denote $\min(\boldsymbol{A}\boldsymbol{z}, \boldsymbol{b})$. Thus, we have:

$$\begin{aligned}
\hat{\boldsymbol{u}} &= \beta^{-1} \left(\boldsymbol{A}\boldsymbol{z} - \min(\boldsymbol{A}\boldsymbol{z}, \boldsymbol{b}) \right) \\
&= \beta^{-1} \max(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{A}\boldsymbol{z}, \boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}) \\
&= \beta^{-1} \max(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}, \boldsymbol{0})
\end{aligned}$$

This is the Euclidean projection into the non-negative orthant, which we will denote $[\cdot]_+$ for compactness. To compute the closed form expression of the smoothed indicator function, we now compute the support function. Using its positive homogeneity property (Beck, 2017, Lemma 2.24.a), we have:

$$\begin{aligned}
\sigma_S(\hat{\boldsymbol{u}}) &= \sigma_S(\beta^{-1}[\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}]_+) \\
&= \beta^{-1}\sigma_S([\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}]_+) \\
&= \beta^{-1} \left(\underbrace{\sigma_{\leq \boldsymbol{0}}\left([\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}]_+\right)}_{=0} - \boldsymbol{b}^\top [\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}]_+ \right)
\end{aligned}$$

Note that we have the following equality:

$$(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b})^\top [\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}]_+ = \|[\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}]_+\|_2^2$$

The smoothed indicator function is then equal to:

$$\begin{aligned}
\delta^{**}_{S,\beta}(\boldsymbol{A}\boldsymbol{z}) &= (\boldsymbol{A}\boldsymbol{z})^\top (\beta^{-1}[\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}]_+) - \beta^{-1}\boldsymbol{b}^\top [\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}]_+ - \frac{\beta}{2}\|\beta^{-1}[\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}]_+\|_2^2 \\
&= \beta^{-1}(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b})^\top [\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}]_+ - \frac{1}{2\beta}\|[\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}]_+\|_2^2 \\
&= \frac{1}{2\beta}\|[\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}]_+\|_2^2
\end{aligned}$$

Thus, we now have a smooth formulation for our optimization problem:

$$\max_{\boldsymbol{z} \in \mathcal{C}} f(\boldsymbol{z}) - \frac{1}{2\beta}\|[\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}]_+\|_2^2$$

Equality case. If the set S is defined by a set of equalities, then we have $S = \{\boldsymbol{b}\}$ and the proximal operator of its indicator function is $\boldsymbol{b}$ (as it is the only element in S). We thus have:

$$\hat{\boldsymbol{u}} = \beta^{-1}(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b})$$

We now compute the support function:

$$\begin{aligned}
\sigma_S(\hat{\boldsymbol{u}}) &= \sigma_S\left(\beta^{-1}(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b})\right) \\
&= \beta^{-1}\sigma_S\left((\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b})\right) \\
&= \beta^{-1}\boldsymbol{b}^\top(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b})
\end{aligned}$$

The smoothed indicator function is then equal to:

$$\begin{aligned}
\delta^{**}_{S,\beta}(\boldsymbol{A}\boldsymbol{z}) &= (\boldsymbol{A}\boldsymbol{z})^\top(\beta^{-1}(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b})) - \beta^{-1}\boldsymbol{b}^\top(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}) - \frac{\beta}{2}\|\beta^{-1}\boldsymbol{b}^\top(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b})\|_2^2 \\
&= \beta^{-1}(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b})^\top(\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}) - \frac{1}{2\beta}\|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}\|_2^2 \\
&= \frac{1}{2\beta}\|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}\|_2^2
\end{aligned}$$

The smooth formulation of our optimization problem is then:

$$\max_{\boldsymbol{z}\in\mathcal{C}} f(\boldsymbol{z}) - \frac{1}{2\beta}\|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}\|_2^2$$

Appendix E

French extended summary

L'analyse sémantique est une tâche qui consiste à produire une représentation formelle manipulable par un ordinateur à partir d'un énoncé en language naturel. Il s'agit d'une tâche majeure dans le traitement automatique des langues avec une large variété d'applications comme le développement de systèmes de question-réponse, la génération de code ou le raisonnement automatique entre autres. Les premiers systèmes d'analyse sémantique reposaient sur des grammaires définies manuellement. Ces approches étaient fortement limitées dans la mesure où elles couvraient des domaines restreints et ne pouvaient pas facilement adaptées à d'autres domaines. Afin de limiter les efforts manuels pour le développement de nouveaux systèmes, des approches se sont intéressées à l'apprentissage automatique à partir de paires composées de phrases associées à leurs représentations sémantiques. Les premiers travaux dans ce but se sont intéressés à l'apprentissage automatique de grammaires qui pouvaient ensuite être employées pour analyser de nouvelles phrases. Ces dernières années, les approches fondées sur les réseaux de neurones, et en particulier les architectures séquence-à-séquence, ont démontré de très bonnes performances pour cette tâche. Cependant, plusieurs travaux ont mis en avant les limites de ces analyseurs sémantiques sur des exemples hors distribution. En particulier, ils échouent lorsque la généralisation compositionnelle est requise. Il est donc essentiel de développer des analyseurs sémantiques qui possèdent de meilleures capacités de composition. La représentation du contenu sémantique est une autre préoccupation lorsque l'on aborde la tâche d'analyse sémantique. Comme différentes structures syntaxiques peuvent être utilisées pour représenter le même contenu sémantique,

il est souhaitable d'utiliser des structures qui peuvent à la fois représenter précisément le contenu sémantique et s'ancrer facilement sur le langage naturel. À ces égards, cette thèse utilise des représentations fondées sur les graphes pour l'analyse sémantique et se concentre sur deux tâches clés. La première concerne l'entrainement des analyseurs sémantiques fondés sur les graphes. Ils doivent apprendre une correspondance entre les différentes parties du graphe sémantique et les mots qui composent l'énoncé en langage naturel. Comme cette correspondance est généralement absente des données d'apprentissage, nous proposons des algorithmes d'apprentissage qui traitent cette correspondance comme une variable latente. La deuxième tâche se concentre sur l'amélioration des capacités de composition des analyseurs sémantiques fondés sur les graphes dans deux contextes différents. Notons que dans la prédiction de graphes, la méthode traditionnelle consiste à prédire d'abord les nœuds, puis les arcs du graphe. Dans le premier contexte, nous supposons que les graphes à prédire sont nécessairement des arborescences et nous proposons un algorithme d'optimisation basé sur le lissage des contraintes et la méthode du gradient conditionnel qui permet de prédire l'ensemble du graphe de manière jointe. Dans le second contexte, nous ne faisons aucune hypothèse quant à la nature des graphes sémantiques. Dans ce cas, nous proposons d'introduire une étape intermédiaire de superétiquetage dans l'algorithme d'inférence. Celle-ci va imposer des contraintes supplémentaires qui devront être respectées durant l'étape de prédiction des arcs. Dans les deux cas, nos contributions peuvent être vues comme l'introduction de contraintes locales supplémentaires pour garantir la validité de la prédiction globale. Expérimentalement, nos contributions améliorent de manière significative les capacités de composition des analyseurs sémantiques fondés sur les graphes et surpassent les approches comparables sur plusieurs jeux de données conçus pour évaluer la généralisation compositionnelle.