THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *4/10/2021* **par :**

Sonia BADENE

Supervision distante pour l'apprentissage de structures discursives dans les conversations multi-locuteurs

---

JURY

Giuseppe CARENINI	Professeur, The University of British Columbia	Rapporteur, Président du Jury
David SCHLANGEN	Professeur, Université de Potsdam	Rapporteur
Farah BENAMARA ZITOUNE	Maitre de conférence, Université Toulouse III-Paul Sabatier, IRIT	Examinatrice
Chloé BRAUD	Chercheuse CNRS, IRIT	Examinatrice
Nicholas ASHER	Directeur de recherches, CNRS, IRIT	Directeur de thèse
Jean-Pierre LORRÉ	Directeur R&D, LINAGORA	Co-encadrant de thèse

---

École doctorale et spécialité :
MITT : Domaine STIC : Intelligence Artificielle
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (UMR 5505)
Directeur(s) de Thèse :
Nicholas ASHER
Rapporteurs :
Giuseppe CARENINI et *David SCHLANGEN*

Abstract

The main objective of this thesis is to improve the automatic capture of semantic information with the goal of modeling and understanding human communication. We have advanced the state of the art in discourse parsing, in particular in the retrieval of discourse structure from chat, in order to implement, at the industrial level, tools to help explore conversations. These include the production of automatic summaries, recommendations, dialogue acts detection, identification of decisions, planning and semantic relations between dialogue acts in order to understand dialogues. In multi-party conversations it is important to not only understand the meaning of a participant's utterance and to whom it is addressed, but also the semantic relations that tie it to other utterances in the conversation and give rise to different conversation threads. An answer must be recognized as an answer to a particular question; an argument, as an argument for or against a proposal under discussion; a disagreement, as the expression of a point of view contrasted with another idea already expressed.

Unfortunately, capturing such information using traditional supervised machine learning methods from quality hand-annotated discourse data is costly and time-consuming, and we do not have nearly enough data to train these machine learning models, much less deep learning models. Another problem is that arguably, no amount of data will be sufficient for machine learning models to learn the semantic characteristics of discourse relations without some expert guidance; the data are simply too sparse. Long distance relations, in which an utterance is semantically connected not to the immediately preceding utterance, but to another utterance from further back in the conversation, are particularly difficult and rare, though often central to comprehension. It is therefore necessary to find a more efficient way to retrieve discourse structures from large corpora of multi-party conversations, such as meeting transcripts or chats. This is one goal this thesis achieves. In addition, we not only wanted to design a model that predicts discourse structure for multi-party conversation without requiring large amounts of hand-annotated data, but also to develop an approach that is transparent and explainable so that it can be modified and improved by experts. The method detailed in this thesis achieves this goal as well.

Keywords: discourse structure, discourse relations, attachment, weak supervision, data programming, computational linguistics.

Résumé

L'objectif principal de cette thèse est d'améliorer l'inférence automatique pour la modélisation et la compréhension des communications humaines. En particulier, le but est de faciliter considérablement l'analyse du discours afin d'implémenter, au niveau industriel, des outils d'aide à l'exploration des conversations. Il s'agit notamment de la production de résumés automatiques, de recommandations, de la détection des actes de dialogue, de l'identification des décisions, de la planification et des relations sémantiques entre les actes de dialogue afin de comprendre les dialogues.

Dans les conversations à plusieurs locuteurs, il est important de comprendre non seulement le sens de l'énoncé d'un locuteur et à qui il s'adresse, mais aussi les relations sémantiques qui le lient aux autres énoncés de la conversation et qui donnent lieu à différents fils de discussion. Une réponse doit être reconnue comme une réponse à une question particulière ; un argument, comme un argument pour ou contre une proposition en cours de discussion ; un désaccord, comme l'expression d'un point de vue contrasté par rapport à une autre idée déjà exprimée.

Malheureusement, les données de discours annotées à la main et de qualités sont coûteuses et prennent du temps, et nous sommes loin d'en avoir assez pour entraîner des modèles d'apprentissage automatique traditionnels, et encore moins des modèles d'apprentissage profond. Il est donc nécessaire de trouver un moyen plus efficace d'annoter en structures discursives de grands corpus de conversations multi-locuteurs, tels que les transcriptions de réunions ou les chats. Un autre problème est qu'aucune quantité de données ne sera suffisante pour permettre aux modèles d'apprentissage automatique d'apprendre les caractéristiques sémantiques des relations discursives sans l'aide d'un expert ; les données sont tout simplement trop rares. Les relations de longue distance, dans lesquelles un énoncé est sémantiquement connecté non pas à l'énoncé qui le précède immédiatement, mais à un autre énoncé plus antérieur/tôt dans la conversation, sont particulièrement difficiles et rares, bien que souvent centrales pour la compréhension. Notre objectif dans cette thèse a donc été non seulement de concevoir un modèle qui prédit la structure du discours pour une conversation multipartite sans nécessiter de grandes quantités de données annotées manuellement, mais

aussi de développer une approche qui soit transparente et explicable afin qu'elle puisse être modifiée et améliorée par des experts.

Mots clés: structure discursive, relations discursives, attachements, supervision distante, programmation par les données, linguistique computationnelle.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor, Prof. Nicholas Asher, Director of Research for the National Center for Scientific Research (CNRS), for the continuous support of my PhD study and research, great motivation, enthusiasm and immense knowledge. His guidance helped me throughout my research and the writing of this thesis. I could not have imagined having a better advisor and mentor for my PhD study.

Besides my advisor, I would like to express my deep and sincere gratitude to my co-author, Catherine Thompson, who carried out this thesis with me, encouraged me and bringing all her knowledge and skills towards completing this work. For those sleepless nights when we were working together to achieve deadlines, attending our first international conference together, being a friendly face in the crowd during those stressful times and sharing the up and down adventures of making a PhD study, I thank her.

Special thanks to Dr. Julie Hunter not only for sharing her ideas and insights on discourse analysis and situated context but also for her patience, her proofreading, useful feedback and guidance throughout the writing process.

Each of these three people have shined a different light on the subject of this thesis and I am very grateful for the broad and unique perspectives on natural language processing and discourse analysis they allowed me to acquire.

I also would like to thank the rest of my thesis committee for their ongoing encouragement, insightful comments, constructive criticism, valuable suggestions and challenging questions. I would like to thank Prof. David Schlangen and Prof. Giuseppe Carenini, for agreeing to review this thesis. I am very honored by the interest they have shown in my work. Their respective works are of considerable importance in my approach to logic and linguistics; I thank them for their time and careful reading. I would also like to thank Prof. Farah Benamara and Dr. Chloé Braud for accepting to evaluate my thesis and sit on my review committee.

My sincere thanks also go to Jean-Pierre Lorré, Bertrand Escudie, Michel-Marie Maudet and Alexandre Zapolsky, for trusting me in the development of this PhD project and placing me on their teams, which led to summer internship opportunities abroad and working on diverse, exciting projects.

I would like to express my thanks to my fellow lab mates in IRIT and in Linagora for the stimulating discussions, their goodwill and for all the fun we have had for the last three years. In particular, I wish to thank the MELODI team leaders Prof. Farah Benamara Zitoune, Prof. Nathalie Aussenac-Gilles and Prof. Philippe Muller for their help, suggestions and for making our work environment so friendly and motivating.

Last but not least, I would like to thank my parents, my sister, every member of my large family and my friends for supporting me spiritually throughout my life, and giving me the curious thirst for knowledge and the open mind that led me to pursue a career in research. *Thanemirth*.

On a final note, I would like to express my appreciation to the French National Research and Technology Agency (ANRT) and the three OpenPaas::NG, LinTO, SUMME-RE projects for providing funding and giving me the opportunity to pursue my thesis.

Contents

Chapter 1

Getting Started

One of the fundamental objectives of discourse analysis is to understand how clauses work together to express complex ideas and form coherent conversations. A speaker usually does not explicitly indicate such relations between sentences; often, an interpreter can naturally deduce them when they listen to a conversation or read a text. For this thesis, we are interested in building computational models of text-level, multi-party discourse interpretation. This chapter will explain how discourse relations and discourse structures are important to support discourse coherence and why building models to infer them constitutes a linguistic, technical and industrial challenge. We will also see that while individual utterances have features that help to indicate which discourse relation is at play in a particular case, none of these features are individually sufficient for inferring discourse structures in general. A computational model of discourse structure must therefore be able to capture how these features work together.

1.1 Motivation

When we listen or read, we naturally make inferences about the connections between individual propositions, or more specifically, *discourse units* (DUs) expressed in the conversation or text, to find the coherent relations[1]; one might provide an explanation for or correction of another, or it might provide an answer to a question, for example. One way of looking at the process of inferring semantic relations is to consider it as an anaphoric process (Chomsky, 1993; Dalrymple, 2005; Hobbs, 1979; Kehler, 1993, 1994). The discourse relations that constitute the complete structure of a conversation or a narration help to reveal the aims of that conversation, such as informing, convincing, manipulating, entertaining, clarifying or

[1] Other terms used to refer to discourse relations are rhetorical relations or coherence relations.

arguing a thesis. It is therefore important to understand how discourse relations are conveyed by the content of utterances in order to follow the continuity of discourse as a whole and, as is important for our industrial uses cases, be able to extract and exploit the most central points in a conversation. Instead of seeing the conversation word by word, or turn by turn, we want to perceive a coherent structure that shows the internal logic of the interactions or the statements.

Discourse relations are often signaled by discourse markers (such as *and, but, because, however* and *although*) (Das et al., 2018; Pitler et al., 2008; Sileo et al., 2019); we classify these as *explicit relations*. But often, the relations are not fully determined by markers (Prasad et al., 2010; Webber, 2016) or are not marked at all with discourse connectives. Consider the following example composed of two sentences, each of which contributes to a single discourse unit:

(1) There are so many people in the shop. There are sales going on right now.

The most straightforward interpretation that comes to mind when reading Example (1) is that there are so many people in the shop *because* of the sales. On this interpretation, there is a discourse relation of explanation that links the two DUs. However, depending on the context, other types of discourse relations between the two sentences could be inferred. For example, the two sentences of (1) might simply list independent events. We could then insert *and* between the two DUs as follows: *There are so many people in the shop, and there are sales going on right now.* In this Example (1), nothing in the content explicitly indicates the right relation between these two DUs, apart from the fact that they are expressed together, one after the other in the same discourse.

When discourse relations are left implicit, other linguistic clues can help guide inferences about how discourse units are related. The order in which discourse units are presented as well as the tense and aspect of verb phrases and anaphoric elements are just some examples. Consider the following pair:

(2) The rain started. A squirrel hurried to its nest.

(3) A squirrel hurried to its nest. The rain started.

In (2), we understand that the rain started and *then* the squirrel headed for its nest. Reversing the order of the sentences, as in (3), suggests the opposite order of events. But now suppose we change the aspect of the second sentence in (3), as in (4):

(4) A squirrel hurried to its nest. The rain had started.

In this example, we understand, as we did in (2), that the squirrel headed to its nest only after the rain started, despite the fact that the sentence describing the squirrel's move comes before the description of the rain; the aspect of a verb phrase influences the temporal interpretation of the eventuality it describes.

Anaphoric elements can also help clarify relations between discourse units, as shown in the following pair:

(5) A man came in. A man sat down at the bar.

(6) A man came in. He sat down at the bar.

The pronoun in (6) indicates that the two discourse units in this example concern the same man and suggest a narrative interpretation of this example, namely that the man first came in and then he sat down at the bar. Such a narrative interpretation is less pronounced for (5): in this case, it is much less clear that reversing the sentences, for example, would impact the overall message conveyed by (5).

While order, tense and aspect, and anaphoric elements can help indicate how we are supposed to understand eventualities described in a discourse as being related, none of these elements is sufficient for determining discourse structure. In fact, the discourse relation inferred between two discourse units can often have a critical impact on temporal and anaphoric interpretation.

(7) John's horse threw him to the ground. He broke his arm.

(8) John broke his arm. His horse threw him to the ground.

(8) reverses the order in which the relevant events are described, but this has no effect on the discourse interpretation; the tendency to understand the horse's throwing John as the cause of his broken arm prevails over the tendency to understand events as happening in the order in which they are described.

Discourse interpretation can also influence the interpretation of anaphoric phenomena.

(9) a. The asteroid hit the Earth.
b. It was destroyed.

Naturally, we interpret the pronoun *It* in (9-b) as referring to the asteroid and not the Earth. World knowledge tells us that it is more likely that the asteroid would be destroyed in this scenario, and in addition, it is statistically the case in grammar that a pronoun in subject position more often refers to the subject of the preceding sentence, rather than the direct object. However, if the following third sentence is added to the discourse: - *c. Humanity has*

no home now., it is obvious now that *It* in (9-b) refers to the Earth and not the asteroid. This is necessary in order to understand how the third sentence coherently relates to the rest of (9). This is what we call *Anaphoric ambiguity* at the pragmatic level; a phrase or word that refers to something previously mentioned, but there is more than one possibility. The result relation between utterances ((9-a) and (9-b)) changed the reference of *It* in the cause-effect relation between [(9-a) and ((9-b) + ((9) c))].

What the foregoing discussion shows is that the way discourse units are understood as relating to each other has a crucial impact on the information conveyed by a discourse. Sometimes, these relations are explicitly encoded by discourse markers such as *because* or *and then*. When they are not, discourse interpretation becomes a very complicated affair that involves, on the one hand, making use of linguistic clues that suggest certain interpretations, and on the other, reasoning about the content of discourse units. A good model for discourse parsing needs to take both into account: the content of discourse units and the form of the sentences that are used to express them.

1.2 Industrial Context

Linguistic interaction plays a crucial role in collaborative work, from the extended conversations we have during professional meetings to interactions required to coordinate on activities such as setting the time for a follow-up meeting. Intelligent Virtual Assistants are becoming more and more common tools for assisting humans in collaborative tasks. They perform simple actions such as sending messages or making appointments, which may require taking into account information about a particular user such as prior commitments scheduled on their calendars. Interaction with these assistants also takes the form of linguistic interaction, but limited to a single user and to a single command; the capabilities of these agents are currently constrained by task-specific frameworks and are based on a very superficial understanding of the user.

Recent advances in Natural Language Processing (NLP) allow for a finer understanding of sentential meaning (Devlin et al., 2018) and also provide models for sentential parsing. However, identifying the discourse structure of dialogues involving two or more participants is a major challenge in computational linguistics. Moreover, taking into account the context of the interactions and relating them to the data associated with dialogue-related activities remains at an embryonic stage (most evaluations focus on sentence pairs (Prasad et al., 2007)).

Let's consider the common situation (10) where we address Siri, Alexa or Google smart assistant.

(10) a. **User:** Ok Google!
b. **User:** Display the files of my Downloads folder.
c. **SmartAssistant:** Downloads folder, coming right up.
d. **User:** Send them all to john@linagora.com.
e. **SmartAssistant:** Here are some results from a search. [displaying several website pages...]

In spite of the impressive improvements in the voice rendering and speech transcription of virtual assistants, a user's intentions are easily misunderstood. As illustrated in (10), problems can arise when a user needs to refer back to something previously said (Schlangen, 2003), giving rise to unsatisfying responses along the lines of (10-e) or even, *Excuse me, I didn't understand. Is there anything else I can do for you?*. These answers are often the result of a dialogue manager whose grammar does not recognize the request. We would like these assistants to understand what we are referring to.

We would also like, for example, that when a user cancels a command to send an email, the assistant does not interpret this move as a dictation of what should be in the email message, as in (11), based on real interactions with Siri assistant. Ideally, it should understand that (11-e) is a correction of the request in (11-a).

(11) a. **User:** Send an email to John.
b. **SmartAssistant:** What's the subject of the email?
c. **User:** "Meeting time".
d. **SmartAssistant:** What would you like your email to say?
e. **User:** Please, cancel the email.
f. **SmartAssistant:** Here's your email message to John ["Please, cancel the email."]. Ready to send it?

Being able to infer discourse relations is crucial for interaction analysis. Finding ways to model the relations that form the coherent structure of a conversation[2] can contribute to improving several aspects of NLP applications. For example, it can help to identify whom a speaker is addressing if there are several participants, highlight important events, find answers to questions asked or retrieve the decisions taken, and even interpret references back to previous utterances. For example, an intelligent assistant could be told to send an email to

[2]While some researchers have argued that ideally each interlocutor should have his or her own model (and hence his or her interpretation of the discourse structure) of a conversation he or she participates in (Asher, 2000; Asher and Lascarides, 1998; Ginzburg, 2012; Venant, 2016), in our project we will assume that all interlocutors eventually reach the same interpretation in the context of a particular discussion and thus share a common discourse model.

all participants of a specific meeting. With Twitter, Facebook, forums, debates and meeting recordings, interactions with virtual assistants or chatbots, strategic interactions during online video games, and so on, there is a growing need for engineers, journalists, and researchers in a variety of disciplines to exploit entire texts of complete conversations.

Linagora[3], a French Small and Medium-sized enterprise (SME) specializing in the development of open source software and collaborative platforms, has a research team dedicated to automatic speech recognition (ASR) and recently, to discourse analysis. One of the most important achievements of the Linagora team is the deployment of conversational speech recognition models. The transcription of oral conversations with several speakers is a very difficult task in advanced ASR systems, requiring large vocabulary language models for transcribing full spontaneous conversations, as opposed to the more limited, closed-domain command models in which command possibilities are predefined with a specific and limited vocabulary. These models need to be able to handle a high level of disfluent speech, which can be difficult for language models trained in grammatically correct speech without hesitations, self-corrections, overlapping speech and other types of conversational movements specific to spoken, multi-party and spontaneous conversation. Other obstacles include difficult recording conditions and recognizing different speaker turns (speaker diarization).

The development of both command models and large-vocabulary ASR models for transcribing full conversations have been a centerpiece of our conversational assistant, LinTO[4], which is designed to allow users to pilot a variety of software and tools with their voices. LinTO is also meant to provide recommendations and task execution to help meetings run smoothly, and also assist in the production of meeting summaries and minutes. These latter tasks have led our team to in-depth study of discourse analysis in tandem with the development of our speech recognition models. Adding to the need for discourse analysis is the recent addition of the collaborative tool, Twake,[5] which works like an IRC chat organized in channels corresponding to as many topics of discussion. The platform also allows the sharing and collaborative editing of files within conversations and integrates within them external services such as GitHub or Dropbox to centralize the monitoring and management of a project. These exchanges create an interest in being able to model the discourse relations not only within chat threads or documents, but higher level connections between different documents and chats and even emails so that important information can be extracted from them.

Building models of discourse structure for spontaneous, multi-party conversation or chat requires corpora specific to this kind of language. As noted above, spoken conversations

[3]https://linagora.com
[4]https://linto.ai
[5]https://twake.app/

can contain a high level of disfluencies that pose problems for discourse parsers trained on well-prepared text. Chat, too, has its own idiosyncrasies: meandering and crossing threads, grammatically incomplete utterances, long threads of discussion, and emojis and GIFs. And often, messages can be edited or deleted without leaving a trace of their original form, which can make it difficult to recover semantic relations between the different utterances.

In addition, machine learning algorithms for discourse parsing generally need to be supervised and thus require *annotated* data, which can be very hard to come by for spontaneous conversation and chat. In collaboration with the MELODI team, from the Toulouse Institute for Research in Computer Science (IRIT)[6], and the company Linagora, we undertook this thesis work to find a way to *automatically predict* the coherent structures of spontaneous multi-party interactions using very little training data or no annotated data in comparison to traditional supervised machine learning and deep learning approaches. Due to a lack of spontaneous, conversational data that is annotated for discourse structure, we decided to focus on textual data, and in particular, chat, as we will explain in more detail in Chapter 3.

1.3 Agenda

We have shown in Section 1.1 that discourse understanding requires inferring relations between discourse units. The structures that are built up from basic discourse units and relations, often very ambiguous, between the different contents of the interactions, help us understand what was intended to be conveyed through the statements. The order in which utterances are made, anaphora resolution, the use of specific connectors, the tenses and aspects of the verbs chosen, and much more, are all ingredients that help us infer discourse relations, but none of them are individually sufficient. The coherent relations that determine the logical structure of a discourse are not always clearly marked or lexically given via discourse connectives; moreover, a given discourse connector can sometimes convey several discourse relations. In addition, even with explicit and unambiguous discourse markers, it can be very difficult to identify the point of attachment of the relation, which is needed to build a determinate discourse structure. We have to find a way to take into account all of these clues, as well as how they interact, in order to predict the structures we want to infer.

The advent of virtual assistants and collaborative platforms has radically changed the way we communicate. In order for these tools to fulfill their potential, however, we need to understand the structure of conversational interactions in the shared environment, how various contributions by various conversational participants are connected together in a coherent whole. However, the lack of annotated data, the noisy nature of spontaneous

[6]https://www.irit.fr/en

conversational data, and the complexities of multi-party dialogues, raise technical challenges that are important to solve.

To address all these areas of concern, we will first of all discuss in Chapter 2 different theories and methods proposed for the analysis and formal modeling of textual discourse and explain why we have chosen Segmented Discourse Representation Theory (SDRT), which we argue is the most suitable for spontaneous, multi-speaker conversations. We will then study in Chapter 3 the specificities of spontaneous conversational data in text format and describe the STAC corpus, the only corpus of multi-party dialogues annotated with full discourse structures, on which we chose to develop and test our model.

In Chapter 4, we will present statistic graphical tools that allow us to annotate and make inferences in the field of uncertainty with distant, disparate knowledge sources. We are interested in these algorithms because there is a real problem in terms of time and means for data annotation, especially when expertise is required. We will present the graphical models representation and how this helps to make inference by establishing dependencies between limited observations. We will then present the Snorkel framework (Ratner et al., 2016) which exploits these graphical representations for the annotation of massive data.

The two chapters that follow (5 and 6) provide a detailed presentation of our experimentation. We will explain how we have used and adapted the discourse theories and tools presented previously to predict coherent discourse structures for dialogues. We will also describe how we incorporated contextual and logical heuristics to best capture and grasp the coherence of interactions as a whole, and carry out the results' analysis at each stage of our experimentation. Our thesis work investigates and demonstrates the potential of applying a weak supervision, data programming approach (Ratner et al., 2016) to the task of learning discourse structure for multi-party dialogue. We assume discourse structures are dependency structures (Li et al., 2014; Muller et al., 2012) and restrict the structure learning problem to predicting edges or attachments between discourse unit (DU) pairs in the dependency graph. Although the problem of attachment is only a part of the overall task of discourse interpretation, it is a difficult problem that serves as a useful benchmark for various approaches to discourse parsing. In addition, we will explain how we use discourse relation types in our model to predict attachment.

Chapter 7 is devoted to a further examination of the results obtained by showing concrete examples of predicted structures. These investigations will show that several predicted interpretations may be correct and we will be able to see how our pre-processing and heuristics directly influence the results.

In the last chapter, Chapter 8, we will distance ourselves from the work carried out in this thesis in order to assess its added value, and to discuss the future work that can be undertaken following this thesis, as well as the benefits of this work at the industrial level.

Chapter 2

Foundations: Theories of Discourse Structure

The examples in Chapter 1 suggest that a discourse will have a structure that is closely related to its meaning. How should we represent these discourse structures formally? Several theories of discourse analysis have been proposed but differ in the way they model the nature of the structures and in the construction rules they propose. In Section 2.1 we will first describe some approaches to modeling structures and constraining their construction. We will then show, in Section 2.2, how dynamic semantic theories, and in particular, Discourse Representation Theory (DRT; Kamp, 1981; Kamp and Reyle, 1993) have been able to create a link between linguistic structure and formal semantics. This will allow us to explain why we chose Segmented Discourse Representation Theory (SDRT; Asher, 1993; Asher and Lascarides, 2003), which builds off of certain aspects of DRT, to represent the attachments between the contents of dialogues.

2.1 Representing Discourse Structures

Discourse theories assume discourse structures to be derived in a process involving three steps. It is necessary to first segment the text into *Elementary Discourse Units* (EDUs), the basic *atoms* of the discourse which can be roughly thought of as clause-level contents.

The second step consists in identifying the attachments between the segments; that is, determining for a given pair of discourse units, whether they are linked by a discourse relation or not. Intuitively, the attachment problem concerns determining, for a given discourse unit, to which units in the preceding discourse it is relevant or semantically related. The final step is to interpret the *type* of attachment. It therefore consists in determining which discourse

relation labels a given positive attachment from the second step. Steps 2 and 3 are closely related but are conceptually distinct. We often can identify an attachment without identifying its relation type—expert annotators might agree that there is a link without agreeing on the type of link. On the other hand, when discourse markers are explicit, annotators might be able to identify the type of relation via which a discourse unit should attach without agreeing on *where* it should be attached. In reality, these steps need not occur consecutively; reasoning about attachments and relation types arguably goes hand in hand.

The segmentation process, crucial in the construction of representations of discourse structure, corresponds to the identification of the minimal spans of text that can serve as arguments to discourse relations. The task definition differs between the different frameworks and has been a subject of research for a long time and still is. Some theories privilege semantic features for isolating minimal spans (Asher and Lascarides, 2003; Hobbs, 1985; Polanyi, 1988), while for others minimal spans are defined by a more pragmatic criterion like a common communicative goal (Grosz and Sidner, 1986; Mann and Thompson, 1988). In both cases, the task of identifying the boundaries of discourse is usually done by paying attention to the syntax, punctuation, key terms and full grammatical construction of the text (Braud et al., 2017b; Joty et al., 2015; Soricut and Marcu, 2003; Sporleder and Lapata, 2005). But it can be said that the granularity of the segmentation is ultimately dictated by the need for consistent relations to achieve the right scope in the modeling purposes of the theory and the specific data annotation project.

The attachment step differs across the various existing formalisms. Each approach defines a methodology for how to attach a new segment of the discourse to the already constructed structure, and for how to form a hierarchical structure. In Rhetorical Structure Theory (RST) (Mann and Thompson, 1988), for example, a discourse unit can only be attached to another unit that is immediately adjacent to it in the discourse, while this constraint does not hold for Segmented Discourse Representation Theory (SDRT) (Asher and Lascarides, 2003), as explained in more detail below. The different approaches to building high level structures then yield different predictions for various linguistic phenomena that depend on discourse structure, such as the resolution of anaphoric expressions.

If the designation of discourse relations varies among authors (Mann and Thompson (1987) talk about *rhetorical relations*, Grosz and Sidner (1986) about *structural relations* and Hobbs (1979) about *coherence relations*), their function is to make it possible to interpret a discourse by linking parts of the text in a coherent way. The main point of divergence concerning coherence relations lies in the set of relations adopted by the theories, which in turn depends on the specific objectives and mechanism of the theory.

What is noticeable in these different definitions of each stage is that even within the same framework, the identification criteria very often differ between the different project annotation guidelines (we will discuss this issue in more detail in section 3.1.1 of the next chapter). We will now examine in more detail the idea of rhetorical relations and the constraints on the construction of discourse structures through two theories; RST, which is the most widely used and SDRT which is more precise from a formal semantic perspective but more difficult to integrate computationally.

2.1.1 Rhetorical Structure Theory (RST)

Rhetorical Structure Theory (RST) (Mann and Thompson, 1987, 1988; Taboada and Mann, 2006) is the most commonly used discourse model in computational linguistics. The standard version of RST, based on Mann and Thompson (1988), adopts a set of 25 rhetorical relations (*extended classical relations*) that can hold between two EDUs (which do not necessarily correspond to basic logical formulae or semantic arguments of relations, but chunks of text in RST) or recursively between larger spans of text in a given discourse.

The graphic representation of a discourse in the RST formalism is a tree which is built by the recursive application of *schemata* in a bottom-up procedure. Each schema application ideally reflects the most plausible relation the writer intended between two contiguous spans of text. The discourse relation inferred then determines the contribution of discourse units to the overall hierarchical structure of the discourse. We explain these points in more detail in the remainder of this subsection.

RST relations

In RST, most relations are *asymmetric*, in that one argument, which is assumed to contain more discourse-essential content, serves as the *nucleus* (or *N*) of the relation and the other, whose content is assumed to be less central and contingent on that of the nucleus, provides the *satellite* (or *S*). Consider the following examples, courtesy of the RST website[1].

(1) Concession(a,b)

a. (S) Tempting as it may be,
b. (N) we shouldn't embrace every popular issue that comes along.

In Example (1) the author's main point seems to be to convince the audience of the content of (1-b); the associated text span is thus taken to be the nucleus of the Concession relation in

[1] https://www.sfu.ca/rst/01intro/definitions.html

(1). The role of (1-a) seems to be to acknowledge that accepting the nuclear content might be a little difficult for the audience. This span thus plays the role of helping the audience to accept the content of (1-b), increasing the audience's positive regard for it. Here, the writer says that he can recognize that it is tempting to embrace (fund) every (this) popular issue, and yet also hold the idea that such ideas (this idea) should not be embraced (funded). (2) provides another example of an asymmetric

(2) Condition(a,b)

a. (N) Employees are urged to complete new beneficiary designation forms for retirement or life insurance benefits
b. (S) whenever there is a change in marital or family status.

The Condition relation in RST recognizes how the content of the nucleus depends on the realization of the satellite. For a full list of nuclear relations in RST, see Mann and Thompson (1988) or the RST website: https://www.sfu.ca/rst/01intro/intro.html.

A small list of relations in RST are *multi-nuclear*, meaning that both arguments have the status of nuclei. They include Conjunction, Contrast, Disjunction, Joint, List, Multinuclear Restatement, and Sequence.

(3) Multi-nuclear Relation: Contrast(a,b)

a. Animals heal,
b. but trees compartmentalize.

This relation has been called Neutral Contrast to reflect the balance of nuclearity, unlike Concession or Antithesis.

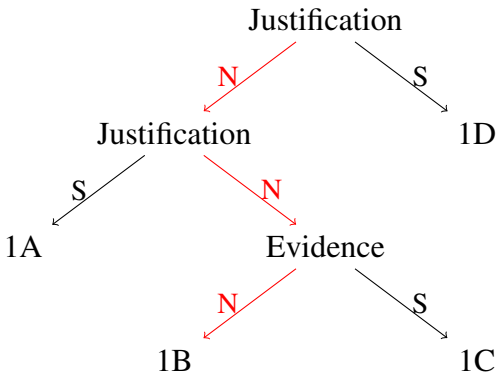
RST structure

RST structures can be represented in a variety of ways. To illustrate, consider the following example from (Marcu, 1996):

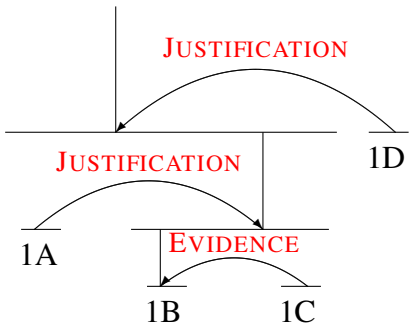
[No matter how much one wants to stay a non-smoker]^{1A} [the truth is that the pressure to smoke in junior high is greater than it will be any other time of one's life.]^{1B} [We know that 3,000 teens start smoking each day,]^{1C} [although it is a fact that 90% of them once thought that smoking was something that they'd never do.]^{1D} (Marcu (1996))

In a standard tree representation, the minimal segments that provide arguments to discourse relations are represented as leaves, while the discourse relation that connects a pair

of segments is represented as a node whose children are either segments (leaves) or other discourse relations. The arcs are labeled with *N* or *S* for *Nucleus* and *Satellite*, as shown in the following graph.



An alternative to this representation is the Schema diagram[2], the most common representation found in the RST literature. These diagrams use curved arrows to indicate nuclearity, where an arrow points to the nucleus. The arrows are then labeled with the name of the relation that connects the satellite of that relation to its nucleus, which furthermore is marked by a vertical line that connects it to its embedding larger segment. (We have chosen option *d* in Figure 4 of the article (Marcu, 1996) to illustrate our explanations, though another option could have been selected.)



One of the major constraints adopted by this model is that a relation can hold between only *adjacent* textual chunks (some exceptions exist with the *Motivation* relation for example where several EDUs motivate one nuclear constituent). If we want to link the segment 1*D* to the segment 1*B* by a *Justification* relation, we will link 1*D* to all the segments formed in the

[2]Schema diagram drawn with RST package created by Reitter (2003a).

tree structure with *Justification*. But there is a problem even if the segments are adjacent but not directly connected to each other. Consider in the example above, if we want to link the segment 1*C* to the segment 1*D* by a *Concession* relation, it is not obvious how to connect them with the chosen structure (option d from Figure 4 of the article (Marcu, 1996)).

Indeed, in the process of annotating a discourse with RST formalism, the main objective after the segmentation of the text into EDUs is finding which adjacent units are to be connected to each other and in which order. These decisions will constrain the hierarchical structure of the tree for the whole discourse. The two following steps in the annotation process are first, labeling attachments with relations and second, specifying the arguments of each relation into nuclear or satellite. Those steps do not constrain the structure of the tree. On the other hand, there may be restrictions on the relations that can be chosen in respect of arguments already tagged as nuclear or satellite. Most discourse approaches ignore distant connections between larger constituents. In RST, rhetorical relations can hold between EDUs or larger textual chunks, given that those larger textual chunks are spanned by one and the same rhetorical relation and the candidate for attachment textual spans are contiguous.

It is easy to see that this RST tree representation is inspired by syntactic trees, which forces the discourse composition from bottom-up. And it is surely for this reason that there is the constraint to bind the adjacent peers of spans together. The relation between span 1B and 1D on the RST tree above is not well visible, this is why Morey et al. (2018); Venant et al. (2013) and Ferracane et al. (2019) have shown how to convert these syntactic structures into dependency trees to better see that the nodes (spans) are the semantic arguments of the relations that contain them. Thus, these dependency-based structures allow relations between non-adjacent utterances and for faster parsers.

The interpretation is ambiguous in relation to the form of the RST structure, but also to the rhetorical relations themselves. They do not depend solely on the linguistic property of the text or segments involved. The 25 asymmetric relations must convey the author's plan from the perspective of the reader. Different RST discourse analyses are therefore possible since it depends on each person's subjectivity, on how he or she understands the writer's intention. For this reason, Moore and Pollack (1992) raised a problem of *intentional structure* of RST representation with regard to semantic and intentional relations. Moore and Pollack (1992) pointed out that RST cannot be used as a means of controlling the structure of discourse in an interactive dialogue system, because RST representations provide insufficient information to support the generation of appropriate responses to *follow-up questions*.

Carlson et al. (2003); Marcu (2000) have demonstrated that the RST concepts are beneficial for automatic text summarization applications and partial discourse parsing. In particular, Marcu (2000) introduced the *nuclearity principle* which states that: a sequence of the nuclear

EDUs of a discourse, connected in a way that reflects the coherent relations, should result in a summary of the text. In the example presented above, the *Justification* relation holds between EDUs 1A–1C (satellite) and 1D (nucleus), therefore it also holds between the nucleus of 1A–1C (i.e., 1B) and 1D. We can therefore summarize the text with *Justification(1B, 1D)*. However, this *nuclearity principle* does not solve the choice of the location of satellite spans, or of nuclei spans in the case of multi-nuclear relations.

Another characteristic of these representations is the single root of the tree from which all other relations are generated. Which forces, in a way, the discourse to have a single objective. We will see in the following section that with SDRT there is no constraint on an additional relation that would come from below the structure to the root.

2.1.2 Segmented Discourse Representation Theory (SDRT)

Before explaining the formalism of SDRT, we will briefly describe the theoretical and methodological foundations of this theory, in order to situate it in the literature and to clarify and justify the choices made with regard to discourse segmentation, hierarchization and the content of the segments. SDRT is a linguistic theory that combines two paradigms of discourse interpretation: dynamic semantics (which we will examine in Section 2.2.3) and discourse analysis. It was introduced by Asher (1993) and Asher and Lascarides (2003) as an extension of Kamp (1981)'s Discourse Representation Theory (DRT) to capture the rhetorical properties at work in discourse. It can be defined as a semantic-pragmatic model for discourse analysis. SDRT provides a consistent and coherent theoretical framework for discourse modeling. It is a dynamic representational theory of discourse that takes into account the segmentation and structural organization of discourse. It aims to describe a deterministic method of constructing Segmented Discourse Representation Structures (SDRSs). Its minimal units are instances of propositions and discourse relations are semantic rather than pragmatically defined in terms of the author's intentions concerning reader's actions as in RST.[3].

SDRT structures (SDRS)

An SDRS is a directed acyclic graph (DAG), $\langle V, E_1, E_2, \ell, \text{Last}\rangle$, where: V is a set of nodes or Discourse Units (DUs) (including both elementary discourse units (EDUs) and complex discourse units (CDUs)); $E_1 \subseteq V^2$ is a set of directed edges between DUs representing coherence relations; $E_2 \subseteq V^2$ represents a dependency relation between DUs; $\ell\colon E_1 \rightarrow R$ is

[3]Other work in dialogue (Asher, 2000; Asher and Lascarides, 1998) aims to extend SDRT to take into account the interaction between intentional structure and semantic or informational structure.

a labeling function that assigns a semantic type to an edge in E_1 from a set R of discourse relation types, and *Last* is a designated element of V giving the last DU relative to textual or temporal order. E_2 is used to represent Complex Discourse Units (CDUs) (Asher et al., 2011), which are clusters of two or more DUs connected as an ensemble to other DUs in the graph. Directed unlabeled edges E_2 connect a complex constituent to its sub-constituents, introducing recursivity in the structure.

The hierarchical structure built in SDRT is obtained by differentiating two types of discourse relations from R: *subordinating* (cf. nucleus-satellite relation in RST) and *coordinating* (cf. multi-nuclear in RST). A *subordinating* relation represents a dominance relationship between the two arguments of the relation. It is typically represented by a vertical arrow ($\rightarrow$). In RST terminology, the dominant argument corresponds to the Nucleus, the dominated argument to the Satellite. Elaboration, Evidence, Explanation, Background and Purpose are subordinating relations. A *coordinating* relation indicates that the two arguments are on equal presentational footing and contribute in the same way to a common dominating topic (Asher and Vieu, 2005). Narration and Continuation are coordinating relations. A coordinating relation is represented by a horizontal arrow ($\downarrow$).

For the narration and dialogue analysis, we can enumerate the following main relations that Asher and Lascarides (2003) proposed:

- **Subordinating:** Elaboration, Acknowledgement, Explanation, Comment, Question-Elaboration, Background, Clarification Question, Question-Answer-Pair.
- **Coordinating:** Narration, Conditional, Alternation, Result, Continuation, Parallel, Contrast, Correction.

Within the category of coordinating relations, we can, following Asher (1993), further distinguish the structural relations Contrast and Parallel, which require their arguments to have similar semantic structure. While the list above provides a basic set of discourse relations, it should be noted that there is no definite list of relations in SDRT. The ones that may be retained are all those whose presence in an SDRS can be shown to modify its conditions of truth (semantic relations) and two relations are distinguished if their contribution to the semantics of the SDRS is distinct.

An SDRS includes two kinds of nodes:

- Atomic nodes π that label EDUs, or the contents of atomic clauses uttered in a particular discourse.
- Primed nodes π' that label CDUs and can thus take atomic labels π and other primed nodes π' in their scope.

Generally, edges that link a constituent to its sub-constituents are dashed in the SDRS representation. Consider the following text example and the corresponding SDRS from (Asher and Lascarides, 2003):

π_1 Max has a great evening last night.

π_2 He had a great meal.

π_3 He ate salmon,

π_4 He devoured lots of cheese.

π_5 He then won a dancing competition.

Figure 2.1 Text example.

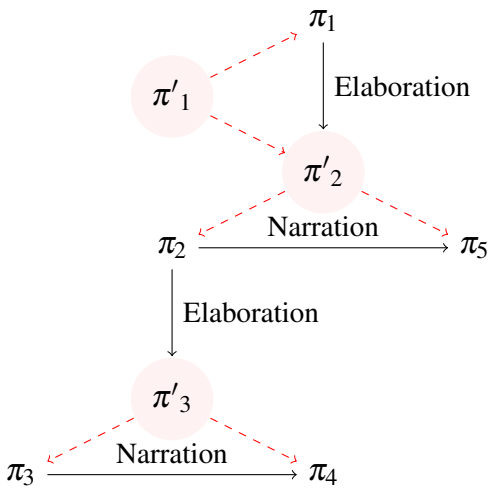


Figure 2.2 The corresponding SDRS.

This example illustrates the role of CDUS. Intuitively, the EDUs π_2-π_5 *together* elaborate on Max's great evening, mentioned in π_1. This elaboration is represented by grouping π_2-π_5 into a CDU, labeled π'_2, that connects to π_1 via Elaboration. Similarly, π_3 and π_4 are grouped into a CDU, π'_3, that elaborates on Max's great meal, introduced in π_2. The entire discourse can also be represented as a CDU, which we label here as π'_1 and whose immediate constituents are π_1 and the CDU π'_2.

The main observation of SDRT discourse representations is the clear understanding of the structure; *"what you see is what you get"*. And one of the main reasons is the construction of CDUs which are built from other DUs. Unlike the Hobbs or Polanyi approaches, the building assembly of the segments is not represented by a tree structure, but directly by the representations of the complex segments, the SDRSs themselves.

SDRT abandons the constraint, adopted in (Polanyi, 1988) and in RST (Mann and Thompson, 1988), that a pair of segments can be linked by only one discourse relation; a pair of EDUs in SDRT might be related by multiple relations. Consider the Example (4), where we have a *contrast* and a *narration* relation connecting (4-a) and (4-b). It is essential here to have these two relations because if we take out *then*, it doesn't have the same meaning; we

need to shift the time forward so they don't overlap, which the contrast relation doesn't do. The marker *but* requires a contrast, but it is not sufficient to produce a coherent example.

(4) a. We used to rent our apartment.
b. But then we bought it.

Even though an SDRS is not a tree, it is an acyclic directed graph with a hierarchical representation where we can determine levels for each constituent by exploiting the different types of relations, coordinating or subordinating (Asher, 1993), that hold between the segments: coordinating ones place their arguments on the same level, while subordinate relations are such that the subordinated constituent is intuitively at a "lower level" than the superordinate one. Also in SDRT CDUs are assigned a level n+1 if their maximal constituent has level n. Thus SDRSs are well-founded structures, and this allows us to do proofs by induction on the complexity of SDRSs.

The hierarchy inferred from the relations affects the available sites for attaching the next segments of the discourse. This is what we call accessibility. To attach a new segment to the already constructed discourse structure, SDRT formalizes a *Right Frontier Constraint (RFC)*, a constraint already suggested (Grosz and Sidner, 1986; Polanyi, 1988; Webber, 1988), according to which not all segments of the discourse structure are available; a new segment can only be attached to the last segment analyzed or to the segments that hierarchically dominate this last segment.

In SDRT, the RFC requires a new EDU to attach to a node along the *Right Frontier* (RF) of the discourse graph to date. The RF includes the last node attached to the discourse graph to date, any node that dominates another node on the RF via a subordinating relation, or any CDU that includes a node on the RF. Apart from rare exceptions, the dominant node of a subordinating relation will precede the subordinate node in a linear ordering (utterance order) of the EDUs in the discourse. If we look again at the graph representation in Figure 2.2, the open attachment sites, i.e. the nodes of the right boundary, correspond to the leaves π_5 and π_1 and to the complex segment π_2' (therefore π_1' as well). The remaining are closed, unless the author explicitly signals the opening of a closed site by means of a linguistic expression (Asher, 1993).

2.1.3 Brief Comparison

The foregoing discussion of RST and SDRT have demonstrated how graph structures can be used to represent discourse structure and how different accounts can take very different approaches. For example, while RST posits tree structures that restrict long -distance

attachments, SDRSs are merely directed, acyclic graphs that allow for a wider range of attachments. With SDRT, we better visualize the semantic arguments of the relations, and the general graph representation allows us to represent more linguistic phenomena. This is not only intuitive for representing the discourses discussed in this chapter, which involve a single speaker, but is particularly suitable for modeling interactions in dialogue.

Dialogue can easily give rise to non-tree-like structures, as when one person addresses several other people, individually, at once, as illustrated in the following example (5):

(5) a. **John:** Do you have a pen?
b. **Paul:** No.
c. **Sarah:** I don't have one.
d. **Lea:** Sorry no.
e. **John:** It's okay, thanks!

In this example, (5-b)-(5-d) do not work together to answer John's question; they happen to all be negative answers but intuitively, they are not sub-parts of one single, coordinated answer. We therefore do not group these EDUs into a CDU. It follows that John's utterance in (5-e) serves to acknowledge multiple, individual EDUs, as indicated in the graph below (where QAP stands for *Question-Answer-Pair* relation and ACK for *Acknowledgement* relation):

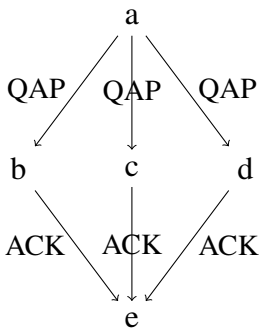


Figure 2.3 The diamond structure corresponding to Example (5).

This diamond shape cannot be represented with a tree. This is a question of representation that distinguishes RST from SDRT, i.e. it motivates the need for general graphs rather than trees because we cannot always have trees as a representation of discourse.

Note that we can reconstruct an SDRS from an RST tree if we assume an appropriate nuclearity principle (Venant et al., 2013). To see this, consider again the example discussed above from (Asher and Lascarides, 2003):

π_1 Max has a great evening last night.

π_2 He had a great meal.

π_3 He ate salmon,

π_4 He devoured lots of cheese.

π_5 He then won a dancing competition.

Figure 2.4 Text example.

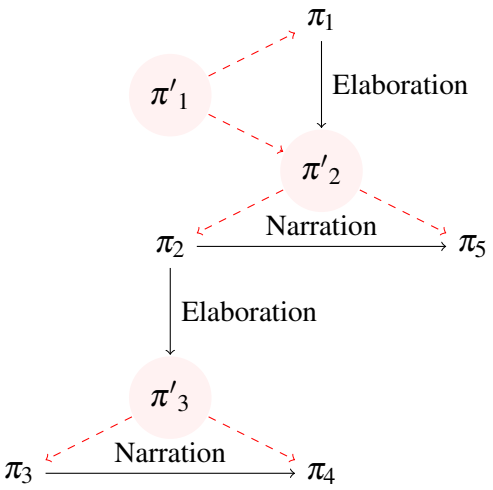


Figure 2.5 The corresponding SDRS.

The corresponding RST structure of the above text is as follows[4]:

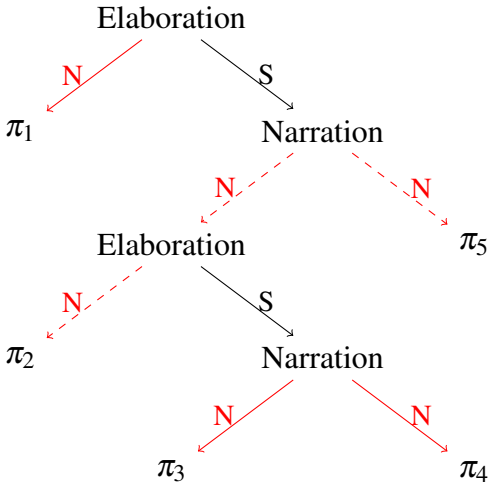


Figure 2.6 The corresponding RST representation.

How do these labels (N, S) help us recompute the SDRT graph from the RST tree? How do they help us determine the semantic arguments of the relations? Let's consider for instance the subtree of the RST structure in Figure 2.6, with a Narration node at its root and spanning

[4]In order to show how to migrate from an RST structure to a SDRS, we have chosen to leave here the names of the SDRT relations, which we could have modified by that of RST, i.e. Narration could have been replaced by List, and Elaboration could have remained as it is (or replaced by Means). We recall that the semantic aspect of relations does not exist in RST (see 2.2.3 for more details).

$\pi_2 - \pi_5$. The path from the root to π_2 has two arcs in it each labeled with *N*. This is a nucleus path and it determines in a dependency graph that π_2 is the first argument of the instance of Narration at the root. The SDRS formula defining the arguments of the Narration root label is thus Narration(π_2,π_5). If we look at the subtree with Elaboration as its root spanning $\pi_2 - \pi_4$, we see that the second *argument* of the Elaboration is a span linked by an arc with an *S*. Following Venant et al. (2013), this means that π_3 and π_4 are grouped together in a CDU π_3 that is the second argument. The formula defining the Elaboration is Elaboration(π_2, π'_3 where π'_3 is defined by the formula Narration(π_3,π_4).

Applying this procedure to the top node of the whole RST structure, we then get the SDRT graph in Figure 2.5. While it is possible to reconstruct an SDRT structure from an RST tree, the opposite is not always possible (Venant et al., 2013).

Both RST trees and SDRSs can be converted into dependency graphs. For RST, this allows for long-distance attachments and better interpretation (Ferracane et al., 2019; Morey et al., 2018; Venant et al., 2013), while in SDRT, for computational reasons, the DAG has always been transformed into a dependency structure because the prediction of CDUs is very complex to compute when taking into account the complete discourse. This transformation replaces any attachment to a CDU by an attachment to the CDU's head *H*, which is simply the textually first EDU within the CDU that has no incoming links. This transformation modifies the interpretations of the structures, i.e. we no longer have an SDRS where *what you see is what you get*. These structures are therefore ambiguous because we cannot tell by looking at them whether a DU is intuitively related to only the head or rather to the whole structure of the *flattened* CDU whose head is *H*.

If expert annotators sometimes think that the discourse is ambiguous, i.e. that there are several possible representational structures that are compatible with what has been said, the logical form of SDRT makes it possible to determine several coherent SDRSs as well. Natural language has ambiguities, whereas the language of SDRT, which is logical and therefore precise, will produce at least two consistent SDRS for the same ambiguous discourse. Other work in SDRT has also shown that it is possible to have a structure per interlocutor when it comes to a discourse of interactions (in multi-party dialogues) (Asher, 2000; Asher and Lascarides, 1998). This allows for a structure of what a participant has understood according to his or her knowledge and context. However, the semantic or informational structure is not dependent on the intentional structure of the interlocutors. In RST, several structures are also possible for the same discourse (Marcu, 1996), but this is not related to the semantics of the discourse, but rather to the schemata in a bottom-up procedure which gives the choice of several options to form the hierarchical RST structure.

2.2 Representation of Meaning

SDRT, in contrast to RST, is a semantic framework. The SDRSs provide a structure or logical form for the discourse, and these structures determine the truth conditions for the discourse. That is, the EDUs in SDRT are semantic contents (instances of propositions) and formulae of the form $R(\alpha, \beta)$, for some discourse relation R and DUs α and β, have clearly defined truth conditions. In this section, we flesh out what it means for content to be semantic in SDRT's sense.

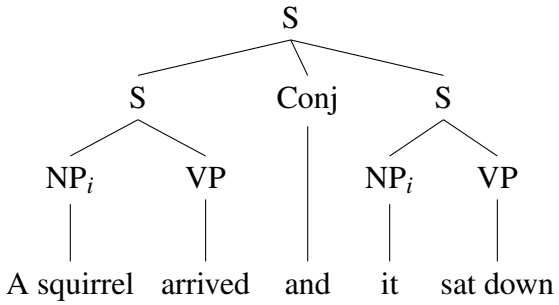
For several years now, formal semantics, which aims to describe and model natural language meaning, have gone beyond the limits of sentences to give rise to dynamic semantics, for instance with Discourse Representation Theory (DRT) introduced by Kamp (1981). Similar works have been exposed in the same period by Heim (1982), Groenendijk and Stokhof (1990). Below, we will review the different approaches to discourse analysis by explaining how formal semantics describes the meaning of discourse evolution. If Montague Grammar (Montague, 1974) lays the foundations of the principle of compositionality to semantically represent the syntax of a discourse, we will not proceed in this Section to a Montagovian analysis, but we will only use first-order logic on simple examples to show how SDRT goes further to represent the semantics of discourse.

2.2.1 Background Concepts

Dynamic semantics developed from a need to model the behavior of anaphoric expressions. Consider (6):

(6) [A squirrel]$_i$ arrived, and [it]$_i$ sat down.

Here, in Example (6), we assume that *a squirrel* and *it* are co-referential (indicated with index *i*). Let's give a simple syntactic structure for this sentence:



We have two sentences, combined together with a conjunction *and* and we have co-indexed the two noun phrases (NPs), *A squirrel* and *it*. If we want to analyze this sentence in terms of semantics, we will get the following formula:

$$\exists x(squirrel(x) \wedge arrived(x) \wedge sat.down(x))$$

We have a proposition that is composed of a quantifier phrase and of conjoined propositions. Each predicate has one argument. The single conjunction allows conjoining any number of propositions. The indefinite determiner *a*, in *a squirrel* in our Example (6), introduces an existential quantifier $\exists$ and a variable x in its scope that serves as an argument to the predicates *squirrel*, *arrived* and *sat.down*.

Now consider Example (7):

(7) [A squirrel]$_i$ saw [a nut]$_j$. He$_i$ ate it$_j$.

This example contains two separate sentences, whose logical forms we represent as follows:

$$\exists x \exists y(squirrel(x) \wedge nut(y) \wedge see(x,y)), \quad ate(\mathbf{x},\mathbf{y})$$

In (7), *it* seems to refer back to the nut introduced in the first sentence, while *he* picks out the squirrel. So long as the logical forms for the two sentences are separate, however, the existential quantifiers introduced by *a squirrel* and *a nut* cannot take scope over the variables x and y, respectively, in the second formula. These variables remain free and we fail to capture the propositional content expressed by the second sentence of (7).

A simple fix for the problem above would be to simply conjoin the formula $ate(\mathbf{x},\mathbf{y})$ to the logical form for the first sentence of (7) and extend the scope of the quantifiers:

$$\exists x \exists y(squirrel(x) \wedge nut(y) \wedge see(x,y), \wedge\ ate(x,y))$$

Unfortunately, such a solution would incorrectly predict that (8) is felicitous, as its content would be equivalent to that of *Every squirrel saw a nut and ate it*.

(8) [Every squirrel]$_i$ saw [a nut]$_j$. He$_i$ ate it$_j$.

In this example, the pronoun *he* in the second sentence fails to refer back to the variable introduced by the quantifier *every squirrel*. Yet if we simply conjoin the content of the two sentences and allow the quantifier to take scope over the content of *he*, we cannot explain this observation.

Another problem with the simple solution proposed above is that it would still fail to prevent the occurrence of free variables in examples like (9):

(9) [Every squirrel]$_i$ who$_i$ saw [a nut]$_j$ ate it$_j$.

We can represent the content of (9) as follows:

$$\forall x((squirrel(x) \wedge \exists y(nut(y) \wedge see(x,y))) \rightarrow ate(x,\mathbf{y}))$$

The variable x is bound by the quantifier $\forall x$, but the variable y remains out of the scope of the quantifier $\exists y$, as the latter is introduced in the antecedent of a conditional. This problem is commonly referred to as *Donkey sentence* (Barker and Shan, 2008; Geach, 1962; Heim, 1990, 1982), and one of the most famous donkey sentences is: *Every farmer that owns a donkey beats it*. A correct translation into first-order logic for the donkey sentence seems to be:

$$\forall x \forall y(farmer(x) \wedge donkey(y) \wedge owns(x,y) \rightarrow beats(x,y))$$

This solves the problem of the free variable in the consequent of the logical form for (9), as y is now in the scope of a universal quantifier. However, it introduces a new problem, namely that the indefinite noun phrase *a* must sometimes be interpreted as contributing an existential quantifier to logical form, and sometimes a universal quantifier.

The above discussion shows that standard quantifier logic doesn't give a satisfactory account for natural language anaphora. We need a dynamic semantics, such as Discourse Representation Theory (DRT) (Kamp, 1981) to resolve these problems.

2.2.2 Discourse Representation Theory (DRT)

The invention of DRT led to a shift from a static to a dynamic view on natural language semantics. Instead of working with first-order formula syntax, DRT makes it possible to work with explicit semantic representations. Its logical forms, Discourse Representation Structures (DRSs), describe the objects mentioned in a discourse and their properties. Let's take (6) again, ([A squirrel]$_i$ arrived, and [it]$_i$ sat down.), to see how DRT would deal with this simple example and analyze its structure.

(10)

x y
squirrel(x)
arrived(x)
y=x
sat.down(y)

The discourse in (6) gives rise to the DRS in (10), a structure in which we have a set of discourse referents U (visually the top box in (10)) together with a set of conditions or first order formulas over the discourse referents in U. Discourse referents translate into first order variables but the construction algorithm of DRT has the effect that these variables have in effect unbound rightward scope unless logical operators intervene.

The construction procedure for DRSs assumes a contextually given DRS (which might be a pair of empty sets for the first sentence in a text) and provides rules for how to formulate the semantic content of each syntactic constituent in a discourse representation structure (DRS). For example, given a contextually given DRS (U,V) when we've got an indefinite *a* like in *a squirrel*, we introduce a new variable x into U, the common noun *squirrel* then introduces a formula on x, $squirrel(x)$, that is introduced into V. *It* is another noun phrase, but because it is a pronoun we will introduce a new variable, here y, that we have to link to a previous referent. In this case, the only previous referent that we've got is x so we are going to link y to x via the condition $y = x$. In DRT, the pronouns require an inference, we have to figure out what is it likely linked to. If we had additional context, the sentence (6) could be interpreted differently where *it* could refer to another referent in the context.

We can easily handle (6) with both traditional logic or DRT. Now let us return to the Example (7) ([A squirrel]$_i$ saw [a nut]$_j$. He$_i$ ate it$_j$.), which we could not resolve with traditional logic because the second proposition "He ate it." had unbound variables. With DRT, we will first create the following structure for the first proposition, "[A squirrel]$_i$ saw [a nut]$_j$.":

(11)

x y
squirrel(x)
nut(y)
saw(x,y)

The predicate *squirrel* is applied to the discourse referent x, and the predicate *nut* to y. And we say that x saw y. Then we get the second sentence, *He$_i$ ate it$_j$*, and we want *He* to be linked to *A squirrel*, and we want *it* to be co-referential with *a nut*. The DRS in (11) is now our contextually given DRS that we update with the material from the new sentence. Thus we expand on our discourse representation. We introduce a discourse referent z for the pronoun *He* and v for the pronoun *it*. We obtain the following extended representation where z is linked back to x and v is linked back to y, and we say that z ate v:

(12)

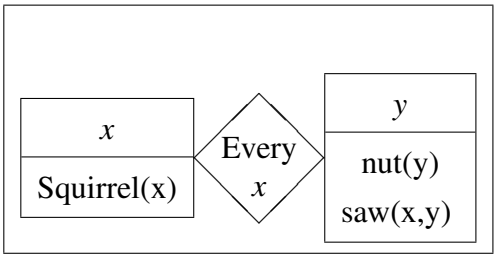
x y z v
squirrel(x)
nut(y)
saw(x,y)
z=x
v=y
ate(z,v)

We now know that it is *The squirrel* that ate *the nut*. This therefore requires inference to associate co-references.

The idea of simply extending the DRS above to accommodate the content of the second sentence of (7) is so far reminiscent of the *simple solution* proposed earlier, in which we conjoin the logical forms of the two sentences in (7) and extend the scope of the existential quantifier over this new conjunct. Let's take a look at Example (8), ([Every squirrel]$_i$ saw [a nut]$_j$. He$_i$ ate it$_j$.), in which we introduce a universal quantifier $\forall$. We had trouble with the *He* in this example because it intuitively cannot be linked to *every squirrel*. In DRT, we start by introducing *a set* of squirrels in our universe of discourse.

(13) [Every squirrel]$_i$ saw [a nut]$_j$.

(14)



Example (13) has a noun phrase, *every squirrel* that gives rise to a universal quantifier. In DRT, *every* introduces a *complex condition* consisting of two DRSs related together by the

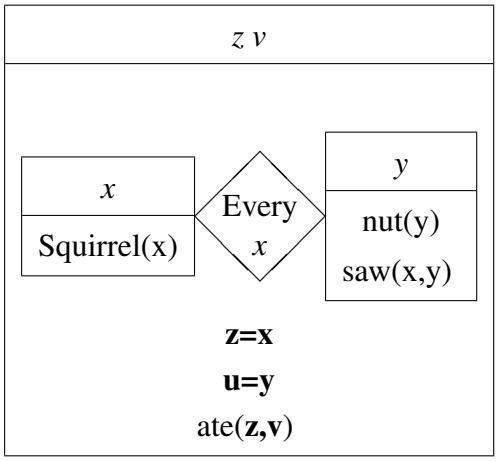
quantifier; in (14) the quantifier is represented by the diamond shape and in the DRS on the left we introduce a variable x and the condition $squirrel(x)$. The DRS construction rules then stipulate that the verb phrase in (13) *saw a nut* introduce material into the DRS on the right. The diamond[5] that connects the left and right DRS says that we're going to look at *every x* in our domain, and for each squirrel, there is a nut and he saw that nut.

We now add the second sentence of (8), namely *He$_i$ ate it$_j$*.:

(15) (8) [Every squirrel]$_i$ saw [a nut]$_j$. He$_i$ ate it$_j$.

The question is where should the material from the second sentence be added? The DRS construction rules stipulate that universal quantifiers have a fixed rightward scope; roughly only the verb phrase (VP) of the first sentence is within the scope of the universal quantifier. Thus the second sentence of (8) must contribute discourse referents and conditions in the main DRS, *not* in the DRS on the right hand side of the quantifier relation (the diamond). So we introduce the discourse referents z and v for the pronouns *He* and *it* in the main DRS or largest box. This leaves us the following structure:

(16)



We notice that neither x nor y is introduced in the discourse domain of the main DRS; they are only introduced in the sub-DRSs. In DRT, this means that the discourse referents x and y are *inaccessible* to the discourse referents z and v, which allows us to predict that the

[5]DRT can also model universal sentences using conditionals. Here it would be, *if a squirrel saw a nut, he ate it*, then there is a squirrel x and a nut y and x saw y in one box, and $eat(u,v)$ in another box where $u = x$ and $v = y$. In the DRS, it would be a simple arrow that would connect the two structures/boxes (Kamp, 2016) in the following manner, where square brackets indicate the limits of boxes [[x, y | squirrel(x), nut(y), saw(x,y)] → [u, v |u=x, v=y, eat(u,v)]]. The discourse referents introduced in the antecedent lie along the accessibility path for u and v, allowing the latter discourse referents to be bound. Note that *a* makes the same contribution as before: it simply introduces a discourse referent. The universal interpretation follows from the semantics of the conditional itself (Kamp, 1981). But only the method described above (with diamonds/lozenge) for representing the universal can be extended to other quantifiers like *many*, *most* or *several*.

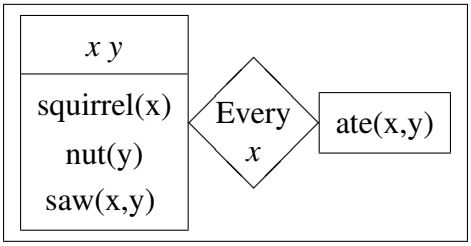
occurrence of *He* will be odd. In more detail, the hierarchical structure of DRSs determines an *accessibility path* through a DRS, such that a discourse referent introduced in one DRS can only be linked to a discourse referent on this path. Roughly, a discourse referent introduced in one DRS can be linked to discourse referents introduced in the same DRS or in a DRS to the left or above. More formally, the accessible domain of a DRS K, is the set of discourse referents that occur in some DRS K' that is accessible to K. K is accessible to K' in DRT if and only if $K = K'$ (every DRS is accessible to itself) or:

1. K contains $\neg K'$
2. K contains $K' \vee K''$
3. K contains $K' \Rightarrow K''$
4. K contains $K'' \Rightarrow K'$
5. there is a K'' that contains $K \Rightarrow K'$
6. there is a K'' that contains $K \diamond K'$
7. K' is accessible to K'', and K'' is accessible to K

The main constraint for the interpretation of the anaphora in DRT is through the accessibility mechanism which stipulates that "A pronoun is represented by a discourse referent x which must be equated to some discourse referent $y \in$ the accessible domain of a DRS K, where K is the DRS in which x is introduced" (Geurts et al., 2020).

Let's take a look at (9), repeated below, to see how DRT manages to represent the content of the discourse. Remember that with first-order formula, we couldn't link the second argument of predicate *ate* to the nut that the squirrel saw: $\forall x((\mathit{squirrel}(x) \wedge \exists y(\mathit{nut}(y) \wedge \mathit{see}(x,y))) \rightarrow \mathit{ate}(x,\mathbf{y}))$. With DRT, it's quite straightforward:

(9) [Every squirrel]$_i$ who$_i$ saw [a nut]$_j$ ate it$_j$.



This DRS first introduces a set of squirrels that saw *a nut*. Then it says that for every squirrel x who saw a nut y, he ate that nut. We see that the accessibility mechanism described above

allows the DRS that contains the predicate *ate* to access *x* and *y* introduced in the left DRS. The idea of DRT is that once we leave the sub-structures, we cannot access *x*, or *y* that *x* saw. We see now how DRT accounts for anaphora.

2.2.3 SDRT, a more elaborate version of DRT

In this section, we will explain why and how SDRT refines the DRT view of semantics. For this, we will take the following examples from (Asher and Lascarides, 2003):

(17) a. Max fell.
b. John helped him up.

(18) a. Max fell.
b. John pushed him.

In (17), we understand that the two utterances happened chronologically in a logical order. John helps Max after falling. There is a *Narration* relation holding between the two chronological events. But in (18) the lexical semantics associated with *fall* and *push* change our understanding of the logical order of events and ensure that information about (18-b) and (18-a) verify a non-monotonic rule from which Explanation(a,b) is inferred. Max fell because John pushed him, or John's pushing Max was *the cause* that Max's falling. There is an *Explanation* relation between (18-a) and (18-b).

Consider the following Example (19):

(19) a. John kicked Max.
b. Max fell.
c. John pushed him.

Here John kicked Max, making or causing Max to fall, and then John pushes him again. We notice that while our understanding of the order of events in (18) was that John's pushing Max was the cause of Max falling, in (19), we already have a cause of Max's fall, which is that John kicked him. Our first understanding of what the cause is blocks us from understanding *John pushed him* as providing the cause. Traditional DRT is not going to be able to account for the different interpretations we get depending on what has come before or comes after. Temporal information in DRT is determined solely by temporal adverbials and verb tense (Altshuler, 2014; Kamp, 1981; Partee, 1973; Partes, 1984). In SDRT this is not the case; rhetorical relations often have an essential role in determining temporal order (Lascarides and Asher, 1993). SDRT helps us interpret this logical order of events. Let's see how we could encode this semantically into a discourse representation structure following SDRT.

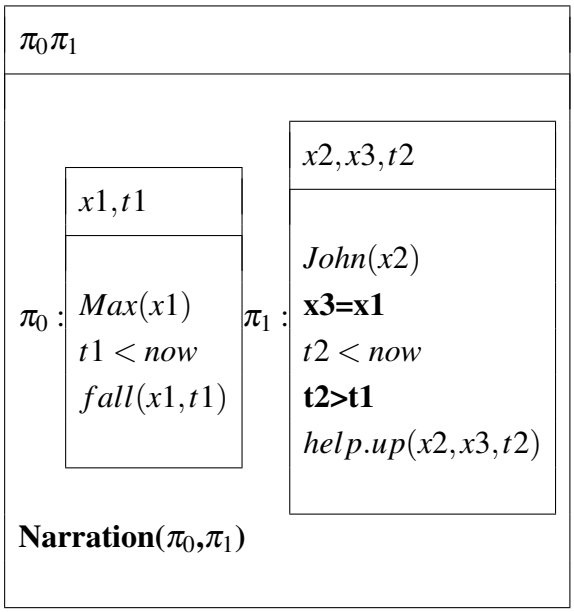
If we encode the first utterance (17-a), we will have the same DRS, except that we add a representation for times or events. So we have the entity $x1$ which we associate with Max, and we have the temporal variable $t1$ tied to the verb, while the tense of the verb says that $t1$ happened before *now/the present*. That is to say that $t1$ is past tense. And we've got our proposition $fall$ that $x1$ fell at time $t1$.

$$\pi_0 : \begin{array}{|l} \hline x1, t1 \\ \hline Max(x1) \\ t1 < now \\ fall(x1, t1) \\ \hline \end{array}$$

By adding the second utterance (17-b), we obtain a more complex structure. But first, let's begin with our preliminary SDRS for (17-b).

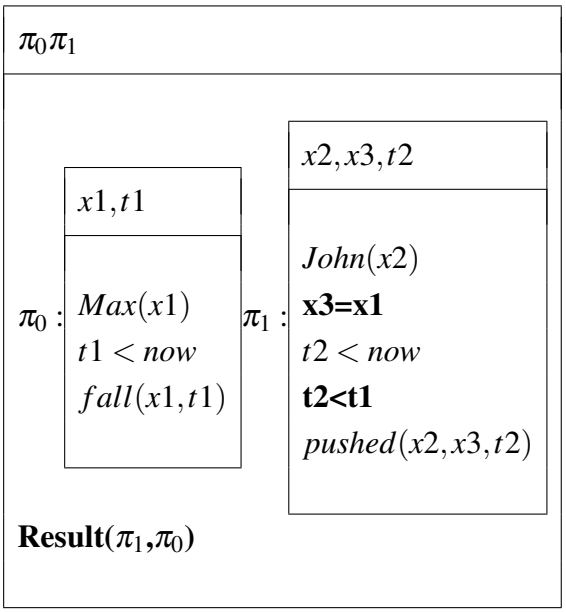
$$\pi_1 : \begin{array}{|l} \hline x2, x3, t2 \\ \hline John(x2) \\ \textbf{x3=?} \\ t2 < now \\ help.up(x2, x3, t2) \\ \hline \end{array}$$

We have now two entities, $x2$ and $x3$. $x2$ is associated with John. $x3$ being a pronoun, should be linked to something. And we've got $t2$ being before now, and $x2$ helped up $x3$ at time $t2$. To understand how to merge the two above structures π_0 and π_1, we need to understand what is the rhetorical relation holding between the two utterances. Here it involves a *narration* relation; Max fell, *then* John helped him up. In SDRT, encoding the relation between clauses, leads to solving the inference for pronouns, specifying tenses and all other phenomena. The complex SDRT structure of (17) is as follows:

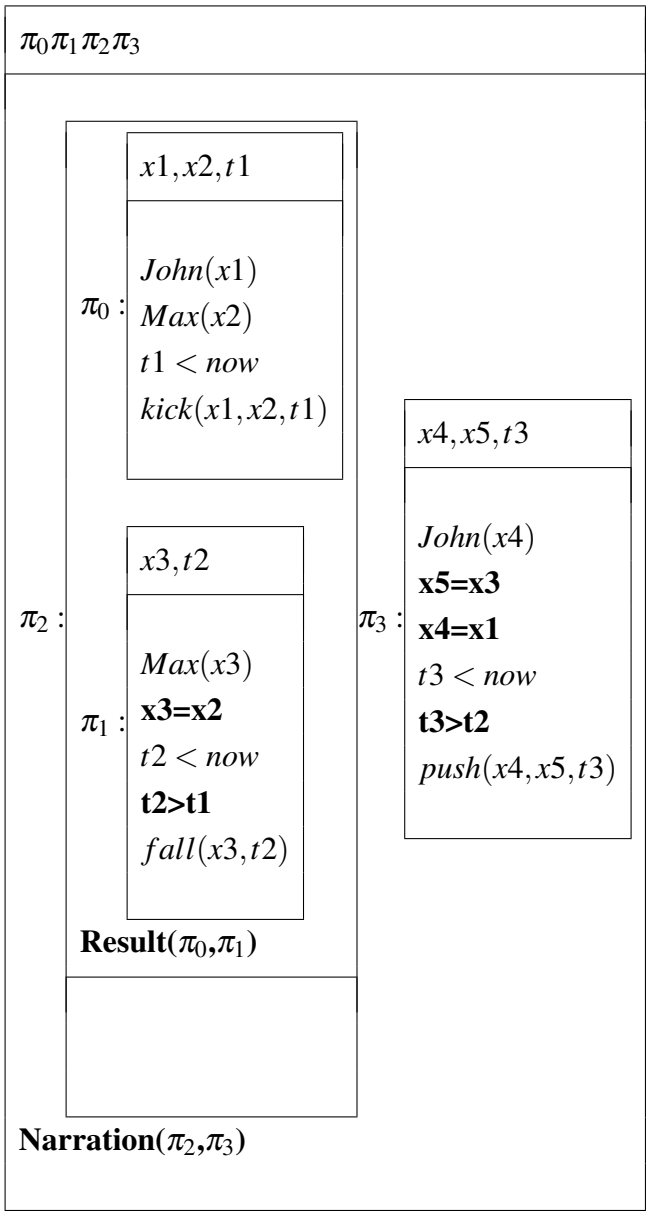


π_0 and π_1 represent the semantic structures of the two utterances (17-a) and (17-b) and we associate these two through *narration* relation. This rhetorical relation is going to tell us that *t2* happens after *t1*. We also solve the inference about *x3* that access the structure π_0 to attach to *x1* thanks to the accessibility rules.

For Example (18), its structure shows once again the importance of rhetorical relations in SDRT for interpreting the discourse. Namely, the cause-effect relation, *Result(π_1,π_0)* will tell us that *t2* precedes *t1*. We also solve the association of the pronoun *him = x3* with *Max = x1*. As with the DRT, the SDRT provides a mechanism for accessibility. This is in line with the explanations of the right frontier (RF) detailed in Section 2.1.2 which constrains the available attachment points in the discourse structure for a new constituent to those of the RF (the last simple constituent introduced in the structure, and any constituent dominating the last one). The discourse referents available for anaphora resolution are those which are DRT-accessible within the constituents of the right frontier from the attachment point. If we look back at our example from (Asher and Lascarides, 2003), a discourse referent X is accessible to a condition in π_4, if X is associated with an SDRS that is associated in π_4, or if it has a direct relation with another SDRS (here a Narration with π_3), or it goes above (we go up the graph and look at the RF) and this CDU π'_3 which contains π_4 is linked to π_2, so we can link π_2, we can also access π'_2 and π_1. This path is called *path of accessibility* in SDRT.



The following structure is the one corresponding to the more complex example (19). π_0 is our first discourse representation for *John kicked Max*. π_1 is the second one for *Max fell*. Here the two first utterances are happening in chronological logical order; we got the cause first and then the effect, therefore it is $Result(\pi_0, \pi_1)$. We assume that the two Max's are the same, but we should encode it in the structure. Because there is still a possibility that we are talking about two different Max's, so if it's not the case, we need to include that inference in the structure. In this example, there is no other context, so $x3 = x2$. Those two structures, connected with the relation *Result*, form the context π_2 for our next discourse utterance. So when we add the representation of the final utterance π_3 of our discourse, it takes as its context the structure π_2. We're going to merge π_2 and π_3 with the *narration* relation; $Narration(\pi_2, \pi_3)$. Because of this connection, $t3$ has to follow $t2$, and we get the other inferences between $x5 = Max$ and $x3$, and that $John = x4$ is the same person in the whole discourse.



Through these examples, we can see why we would like to use SDRT and not another approach. Other approaches cannot explain the logical order of events and other phenomena of language. In SDRT, the rhetorical relations are semantic functions for the interpretation of discourse. It defines what is going to be a possible and not possible connection between semantic representations. SDRT is therefore a very promising approach for accurately encoding linguistic phenomena.

2.3 Conclusion

Human languages have a wide variety of features that are used to convey meaning. Among the most important of these is the ability to convey a predicate-argument structure. First-order logic offers this meaning representation; however, we have seen that it is quite limited for representation of discourse in its context and dynamism. Our principal objective in this chapter has been to introduce some of the main approaches to building representations of discourse structure, with a special focus on RST and SDRT, by looking first at the mechanisms of representation and then at the semantic side of SDRT (RST is unclear with regard to the semantic interpretation of its rhetorical relations and does not offer a precise logical framework). SDRT provides a language for representing the logical form of discourse and of dialogue. An SDRS, which is the structure of a coherent discourse, is a discourse unit consisting of rhetorically connected discourse units. Meaning representations for discourse in SDRT have a dynamic semantic interpretation that capture, through discourse relations, additional content to that given by compositional or lexical semantics of the utterances they link together to form a complete coherent structure.

We have seen that theories of discourse structure differ (mainly) with respect to their definition of discourse relations, the arguments (semantic or not) of these relations, the hierarchical constitution, thus also on different theoretical foundations, different aims, and different tasks for which they are appropriate (or not). The general graphical form of SDRS also makes it possible to represent linguistic phenomena, in particular diamond structure for multi-party dialogues, which RST does not allow. Venant et al. (2013) have shown that SDRT can integrate the representation of RST in a dependency-form, but not vice versa. It is therefore natural to choose SDRT as the theoretical basis for an analysis of multi-party conversations we want to address.

In the following chapter, which is dedicated to data, we will identify the features specific to the data we want to analyze in order to review our insights into discourse theories. We will also highlight a major problem for computational approaches to predicting discourse structure (and even more generally on any type of data, as for images, or audio), namely the annotation of data. Even within a single framework, data annotation is often a difficult task, both in terms of the time it takes to annotate the data needed and in terms of disagreement on the choice of structures.

Chapter 3

Foundations: Dialogue and Data sets

Our project is about predicting discourse structure for spontaneous, multi-party conversations, and this chapter is devoted to the issue of data. We need data to build a model of conversational discourse structure in order to strengthen or refute our intuitions. Science has always been about collecting empirical data which experts analyze and then model. Recently, data driven methods have promised automatically generated models. The catch is that they can do this when there is enough data and of the right kind. Yet adequate discourse data is scarce.

We will structure this chapter around two objectives. First of all, we will use real data to highlight features particular to spontaneous multi-party dialogue, through the presentation of the STrAtegicConversation (STAC) project, on whose corpus we conducted our experiments. As we will see, these specific features complicate the prediction of attachment. In the second part, we will present recent efforts undertaken for the development of data-driven discourse parsing and what the success of that has been. We will highlight the difficulties encountered in the process of annotating data as well as the challenges this presents for computational approaches.

3.1 Conversational Data (Spontaneous + Multi-party)

As explained in Chapter 1, monologue can leave certain ambiguities to be resolved in the process of interpretation, but when a discourse involves several speakers, the game of inference can become even richer and more sophisticated. Let's take the following example of a conversation with three people:

(1) a. **Tom:** How much are these flowers?
b. **Jessica:** Eleven euros a bouquet.
c. **Sarah:** Do you have white tulips?

d. **Tom:** Why do you want white ones?
e. **Jessica:** No, sorry.
f. **Sarah:** (i) Never mind. (ii) We'll buy those red ones.

Several conclusions can be drawn from (1). For example, we can ask where this conversation takes place and who Jessica is. The words *flowers* and *bouquet* together with the fact that the speakers talk about prices suggest the conversation probably took place in a florist's shop, and not in a garden. But more importantly, we can ask ourselves questions to understand what is being said and why. To whom is each conversational contribution addressed? To one person in particular, or to all the conversational participants? Are (1-d) and (1-e) both reactions to Sarah's question (1-c)? Does *Never mind* in (1-f) refer to Jessica's *no sorry* or to the question in (1-d)? Is clause (ii) in (1-f) a Result only of Jessica's answer (1-e) to Sarah's question in (1-c), or to both Jessica's answer (1-e) and Tom's intervention in (1-d)? Sarah's clause (ii) in (1-f) seems to be a cancellation of her implicit request in (1-c). To understand this conversation, it is thus important to know who is speaking to whom.

(1) shows that modeling dialogue involves taking into consideration different kinds of cues in addition to the ones encountered in monologue. To shed further light on the particular nature of dialogue, we will present the STAC corpus of situated chats—the corpus that we used for our experiments—and walk through some concrete examples from this corpus. First, however, we begin with a brief overview of corpora that have been used to study the discourse structure of monologues.

3.1.1 Data sets

Quite a few discourse annotated corpora now exist for single authored text—most notably the Penn Discourse Treebank (PDTB) (Miltsakaki et al., 2004; Prasad et al., 2007) the RST Discourse Treebank (RST-DT) corpus (Carlson et al., 2003, 2002) with full discourse structure for texts, as well as smaller annotated corpora such as DISCOR (Baldridge et al., 2007) or ANNODIS (Afantenos et al., 2012a). Recent years have seen the emergence of corpora for segmentation, prediction of discourse relations and more seldom (or less straightforwardly[1]) prediction of complete discourse structures. The study of discourse structure in frameworks such as RST, and SDRT, has seen a revival in recent years, and several researchers are now actively engaged in the creation of discourse data; we can mention The Potsdam Commentary Corpus (Stede and Neumann, 2014), The Georgetown University Multilayer (GUM) corpus (Zeldes, 2017), The RST Discourse Treebank (Carlson

[1]Full structure prediction is often achieved by reconstructing the entire dependency graph from the relations and their directionality or the hierarchical frameworks they follow when this is possible.

et al., 2002), Dutch Discourse Treebank (NLDT) (Redeker et al., 2012), Cross-document Structure Theory News Corpus (Cardoso et al., 2011), Russian RST Treebank (Pisarevskaya et al., 2017; Toldova et al., 2017), The RST Spanish Treebank (Da Cunha et al., 2011), The RST Spanish-Chinese Treebank (Cao et al., 2016, 2017a, 2018, 2017b), The Penn Discourse Treebank (PDTB) (Prasad et al., 2018), Chinese Discourse Treebank 0.5 (Zhou et al., 2014), DISCOR project (Discourse Structure and Coreference Resolution (Baldridge et al., 2007; Reese et al., 2007), ANNODIS (Afantenos et al., 2012a), and the STAC corpus (Afantenos et al., 2012b; Asher et al., 2016).

While several discourse annotated corpora exist for single authored text, there are very few discourse annotated corpora of multi-party interactions. In fact we know of only one, the STAC corpus (Afantenos et al., 2012b; Asher et al., 2016), which provides full discourse annotations on a corpus of chats from an online version of the game Settlers of Catan. Other corpora of multi-party interactions exist, including meetings (e.g. ICSI (Morgan et al., 2001)[2] and AMI (McCowan et al., 2005)),[3] telephone conversations (Zhou et al., 2010), and emails (Joty et al., 2011)), but they are either not annotated (raw data collection) or are not annotated for full discourse structures. As STAC is the only corpus of multi-party dialogues annotated with complete discourse structures following SDRT, this is the one we used in our work. This corpus has two versions, one that has only the linguistic contributions that participants made to the chat threads (Afantenos et al., 2012b), and another version that contains descriptions of actions in the game, temporally ordered relative to the chat moves (Hunter et al., 2015). These two corpora will allow us to transfer and evaluate our approach.

As mentioned in our industrial motivation in Chapter 1.2, we want to analyze and annotate transcripts of meetings and messages in corporate chat platforms. The major difference between oral meetings and chats is that even if both have spontaneous interactions, oral meetings are synchronous and full of disfluencies such as filled pauses, stutters, self corrections in mid-sentence, repetitions and false starts. One of the results of having disfluencies is that it is hard to recover a continuous line of thought from spoken discourse (in comparison to well-prepared written monologues). Instant messages such as chats are asynchronous with reactive planning, corrections of discourse sequences, and meandering threads. While we don't have disfluencies in STAC, we have a similar problem of participants constructing the discourse as they go along and getting off-topic and reformulating their discourse moves as a reaction to other participants.

[2]https://groups.inf.ed.ac.uk/ami/icsi/
[3]https://groups.inf.ed.ac.uk/ami/corpus

The STAC corpus is therefore a good alternative to meeting transcripts given the current state of conversational corpora, in order to develop and evaluate a dialogue model. We describe the STAC corpus in more detail in the following section.

3.1.2 Strategic Conversation (STAC)

The STrAtegic Conversation (STAC) corpus is a collection of chats between three or four players of an online version of the game *The Settlers of Catan*. The rules of the game[4] allow us to have a common context that can help us to interpret the linguistic chats. Players use resources to build roads, settlements and cities on a game board (see Figure 3.1), the Island of Catan, that consists of 19 terrain tiles (or hexes). The goal is to expand territory and be the first player to reach 10 victory points (each settlement is worth 1 victory point, and each city is worth 2 victory points) and thus win the game. They can acquire resources by trading with other players or with the bank, by playing special cards, or by rolling two dice when it is their turn. Each turn, the player rolls the dice, and the number rolled determines which terrain hexes will yield resources. Each terrain hex is associated with a type of resource and marked with a number between 2 and 12 (corresponding to the possible outcomes of the dice rolls). If, for example, a 4 is rolled, all terrain hexes marked with a 4 will yield resources to players that own a settlement or city on such a hex. The type and quantity of the resource thus distributed depends on the resource associated with the hex. In Figure 3.1, for example, a roll of 11 will distribute brick (red/brown) to Joel, who has a (white) settlement on a brick hex marked with 11 (in the center). Rolling a 7 initiates a special series of actions: the current player must move a game piece called *the robber* to a hex of his choice, and then steal a resource from a player with a building on that hex. The robber will stay on that hex until it is moved in another turn, and its presence will continue to have an impact on the game by blocking the distribution of resources for the occupied hex. These rules provide background knowledge that roughly models the sort of background knowledge that one might have about meetings or other conversational situations.

Players interact via a chat window (*Chat* in the board game showed in Figure 3.1) and can see the chat history in the window *History*. The game history, shown in the *Game* window, provides descriptions of the principal extra-linguistic events from the game (e.g. dice rolls, resource distributions, building events, card plays). Messages from the Game and History windows, along with a record of every change made to the game board or to a player's hand, were recorded in User Interface (UI) logs. The game logs also automatically provide

[4] https://www.catan.com/service/game-rules

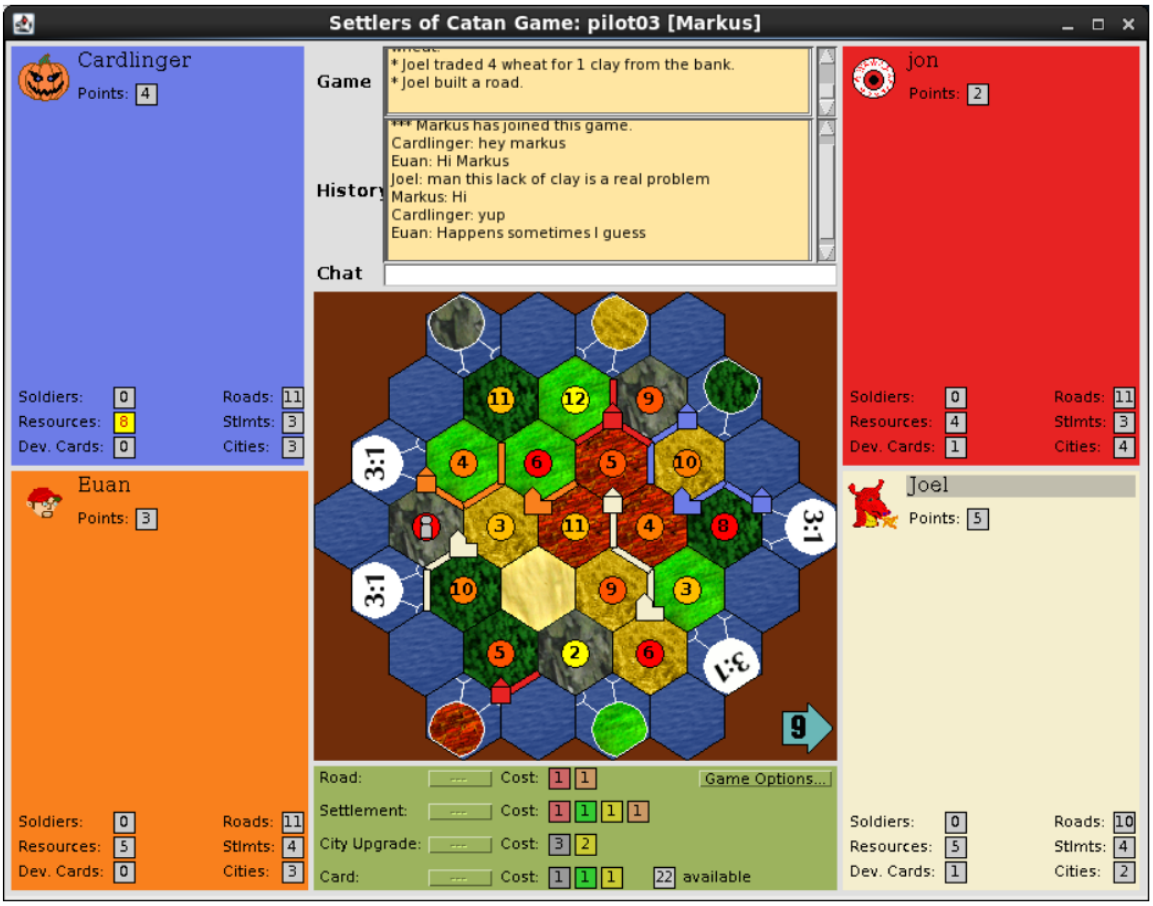


Figure 3.1 The online interface of the Settlers of Catan game as seen from a single player's point of view.

information such as the timestamp for each linguistic and extra-linguistic moves, and the moves' emitters (player pseudonym, Server or UI).

The STAC corpus was annotated in two versions: the *linguistic only* version (or non situated) and the *situated* version. During the first phase, relations were added only to the linguistic chats between players from the Chat/History windows (these player utterances are the *linguistic* conversational turns). During the second phase, descriptions of the game moves were added in the form of *non-linguistic* units, which we call *elementary event units* or EEUs, in order to *situate* the linguistic content. This situated corpus is built with descriptions from the Game window (Server messages) together with certain descriptions that had to be recovered from the game (UI) logs (Asher et al., 2016; Hunter et al., 2015, 2018).

Data from recent work on grounded agreement games (Attari et al., 2019; Chattopadhyay et al., 2017; Das et al., 2017; Ilinykh et al., 2019; Kottur et al., 2018; Schlangen, 2019; Schlangen et al., 2018; Skantze, 2005) shows some similarity with the STAC corpus; for

example, there are, arguably, discourse relations between the messages exchanged by players, and actions performed in the game. In both STAC and grounded agreement games, players exploit visual/non-visual (non-linguistic information from UI logs + Server messages in the Game window) and linguistic information. However, in STAC unlike the work of (visual) Conversational Grounding players only rarely need to coordinate to understand what is going on. In part this is because they have a common background knowledge of how to play the game and also their virtual environment has only one event to attend to at a time.

The STAC data-set was developed within the context of the STAC (STrAtegic Conversation) project supported by the European Research Council, Grant n. 269427. The STAC project sought to provide a rich annotated corpus for analyzing multi-agent strategic conversations. As such, each conversation typically involves a series of negotiations to reach an agreement or to close a deal in the game. The conversations are spontaneous, and therefore also offer opportunities for discussion outside the context of the negotiations. Figure 3.2 shows an example excerpt from the STAC corpus.

177.0.2	Server	Its Cats turn to roll the dice
178	Server	Cat rolled a 5 and a 5
179	Server	william gets 2 sheep
179.0.0.1	UI	Cat has 6 resources. Thomas has 11 resources. william has 6 resources.
180	Cat	anyone would like to give me an ore?
181	william	cant
182	Thomas	what for?
183	Cat	wood?
184	Thomas	no thanks
184.0.1	UI	Cat ended their turn

Figure 3.2 Example of a dialogue from the STAC corpus.

The resulting data-set consists of 45 games. Each game was then divided into what were called *dialogues*, allowing us to increase the number of data structures from which to learn. In the linguistic annotations, the breakdown of the games into dialogues was done by

negotiation sessions (trade negotiations) but there are cases where there were links between negotiation sessions (a discourse relation that clearly linked two dialogues), in which case the annotators combined the negotiation sessions into one. In the situated annotations, game moves had to be taken into account. The annotators then decided to separate the dialogues by rolls of the dice (turns in the game). Each negotiation is properly included in a game turn, though a game turn that can include more than one negotiation. A dialogue thus begins with a roll of the dice by a particular person who then continues until they perform the non-linguistic action in the game of ending their turn (see example from Figure 3.2). In the interim, players can bargain with each other or make spontaneous conversation. The players do not all stop conversing until the end of the game and they sign off. There are typically many such conversations, each beginning with a non-linguistic turn in which a player is designated to begin negotiations.

The annotation process has several levels in STAC. First, each dialogue is segmented into Elementary Discourse Units (EDUs) and EEUs. Chat turns, which are automatically delimited in the History window, provide a lot of information about EDU segmentation, as a single EDU will never span two chat turns with different authors; however, some chat turns needed to be further broken down for EDU segmentation. The example in Figure 3.2 shows a complex chat turn, 184, involving two EDUs, [no] and [thanks], that serve distinct discourse functions. The EDU [no] is an answer to both Cat's questions in turns 180 and 183, and the clause [thanks] is a comment on Thomas' answer *no*. Descriptions of game events must also be further segmented to attain EEUs. Turn 179.0.0.1 in Example 3.2 is therefore divided into three EEUs; [Cat has 6 resources.], [Thomas has 11 resources.] and [william has 6 resources.].

After segmentation, each EDU or EEU was then annotated with labels for:

- *Surface acts* which are types of speech act: QUESTION, REQUEST, or ASSERTION.
- *Dialogue acts* that are more specific to the game: OFFER, COUNTEROFFER, ACCEPT, REFUSAL, or OTHER.
- The *addressee*: for each EDU or trade move in the game, annotators marked the addressee for that move, where the addressee could be a single player, all the (non-speaker) players, or some proper subgroup of the other players.

A third annotation process required identifying the discourse relations between EDUs and, for the situated annotations, between EDUs and EEUs (which, together with CDUs, we collectively refer to as DUs, or discourse units). This required solving two problems: the *attachment problem*, which is finding to which DUs in the constructed SDRS an EDU/EEU

Result	The source DU describes the cause of the eventuality described by the target DU.
Acknowledgment	The target DU signals an understanding or an acceptance of what was said in the DU to which it is attached.
Continuation	The target DU continues a discourse function of the source DU and shares a topic with the source DU.
Elaboration	The target DU provides more information about the eventuality described by the first DU.
Conditional	The source DU is a hypothesis and the target DU is a consequence of the hypothesis.
Contrast	The two arguments are contrasting, meaning they are compared in such a way as to emphasize the striking differences.
Question answer pair	The source DU provides a question, and the target, an answer to that question.
Q-elab	The target DU provides a question that elaborates on a question provided by the source DU.
Narration	The target DU describes an event that occurs after that described by the source.
Correction	The target DU provides a correction of something said in the source DU. The target DU can negate the content of the source DU.
Explanation	The target DU describes the cause of an eventuality described by the source DU.
Alternation	It marks a disjunction between the two DUs.
Parallel	Links two DUs whose contents are similar or analogous; the eventualities described are often taken to be simultaneous.
Comment	The target DU provides an opinion or an evaluation of the content associated with the source DU.
Clarification Question	The target DU provides a question that tries to clarify the content of the source DU.
Background	The target DU provides a staging for the eventuality described by the source DU.

Table 3.1 List of rhetorical relations used in the STAC corpus.

is attached as an argument of a discourse relation, and the *labeling problem*, which involves labeling discourse attachments with labels for discourse relations. In the STAC corpus there are 16 discourse relations displayed in Table 3.1, and they represent the rhetorical functions between the EDUs and EEUs.

Annotations for dialogue act information and discourse relations were performed with the Glozz tool[5]. The annotations benefited from several passes—a first pass done by annotators hired for the STAC project and subsequent revisions carried out by SDRT experts. Complete discourse structures were built for each dialogue. Figure 3.4 shows the output structure of the dialogue example in Figure 3.3. For visualizations of the discourse structures for all of the dialogues in the STAC corpus, see the STAC website (www.irit.fr/STAC). In these graphs the dots represent the segmented DUs (black for EDUs, yellow for EEUs and red for CDUs) and the colored arrows indicate the different discourse relations.

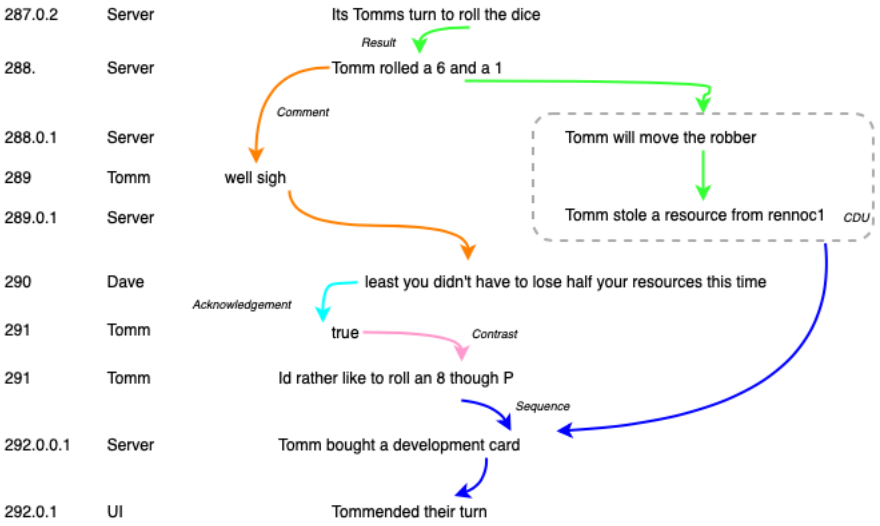


Figure 3.3 Example of a dialogue structure in STAC.

3.1.3 Understanding the Dialogue's Dynamics

If the analysis of monologue showed a dynamic formalization, the dynamic aspect is more visible when we are in the presence of a dialogue involving two or more participants. We will outline some concrete examples specific to dialogues and to our corpus, and we will illustrate how these aspects represent additional problems for discourse attachment.

Non-treelike Structures in Dialogue

One of the particularities of the multi-party dialogue, which is less common, or even absent in the monologue, is the presence of non tree-like structures. One such example is the *diamond* structure (Afantenos et al., 2015; Asher et al., 2016; Perret et al., 2016) that was discussed in Chapter 2. Diamond structures are a special kind of non-treelike structure, which can be

[5]http://www.glozz.org

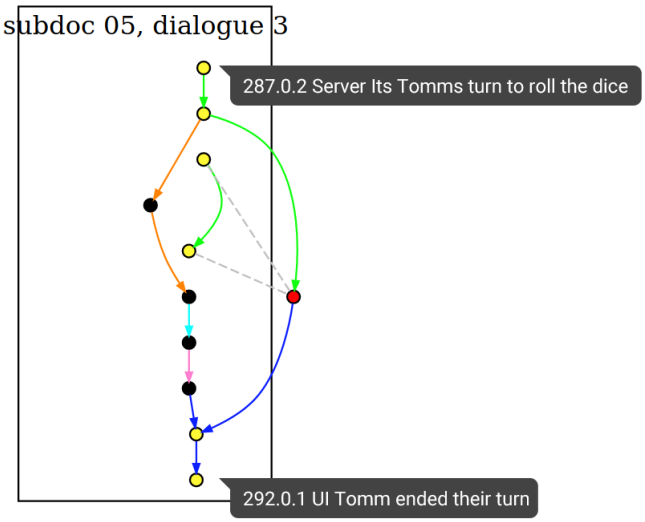


Figure 3.4 Diagram showing a dialogue box from the project website where the first turn is from the Server and the last one from the User Interface (UI) which indicate the initial movement of Tomm and when he ends his turn.

called *truly non-treelike*, that is indeed not seen in monologue. This structure is characterized by a DU that has two or more incoming arrows from *different* source DUs. In Figure 3.5 (Graph visualization link), in turn 234, GWFS makes an offer. After receiving three negative responses (235, 236, 238), GWFS acknowledges all three responses at once in turn 239. Turn 239 has three incoming arrows from different source DUs.

Figure 3.5 A diamond structure from the STAC corpus.

Long Distance Attachments to Very Long Distance Attachments

One of the primary ways in which to analyze the structure of dialogues is through the idea of the *conversational turn*. Because there are two or more agents (multi-party) in

...		
0.0.6	UI	Chameleon sat down at seat 1
1	skinnylinny	Hello Chameleon
1.0.1	UI	Nancy joined the game
1.0.2	UI	Nancy sat down at seat 3
2	skinnylinny	Hello Nancy
3	Nancy	Hi
4	Chameleon	Hi Skinnylinny Hi Nancy
5	Nancy	Hi Chameleon
5.1	Marcus	OK you 're ready to go. You 'll be only 3 in this game.
6	skinnylinny	Ahh ok
6.0.1	UI	Game started
...		

Figure 3.6 Greetings before the game starts.

dialogues, they typically have turns. But these conversational turns do not determine much about the structure without the discourse relations that explain what is going on within each participant's contribution to the turn. Turn taking instantiate one or more discourse relations between some constituent and available attachment site. Every conversational turn can contain one or more EDUs, but one EDU cannot be contained in several conversational turns. We need explicit cues to pair up the content of different DUs but as we saw it in Chapter 1, even with explicit signals, it is often unclear and ambiguous what a constituent attaches to.

An indication to the attachment would be the *pairwise organization* of multi-party conversations. The interactions can be broken down into a series of 2-utterance pairing called *adjacency pairs* (which are not an explicative structure but only an indication pattern), e.g. (command/acceptance), (greeting/counter-greeting) and (query/reply). Adjacency pairs are two turns uttered by different speakers, one in response to the other. These simplified patterns are observed in the STAC corpus as well; e.g. a polar question which is attached to the expected yes-no answer, or X who greets Y followed by Y who greets X in turn. An example from STAC in Figure 3.6 (Click here for full graph visualization) illustrates simple

attachments for greetings. These expectations of specific type of move, which are often called *direct relevance*, of what is going to be said after what has been said, have been studied a lot, especially for the question/answer relations (Asher and Lascarides, 1998; Carlson, 1984; Ginzburg, 2016; Walton and Macagno, 2017). Groups of expressions which often go together can then be formed to link two discourse units. However, this concept is very simplified for the spontaneous multi-party conversations we want to analyze. There can be several DUs in one conversational turn with different dialogue acts. Thus in a following turn there might be a response to a unit in the first turn, but also other material. Figure 3.6 illustrates a more complex link between EDUs in different conversational turns, e.g. between Markus EDU [you're ready to go] and the EDU [ok] from skinnylinny in turn 6.

Adjacency pairs are frequently interrupted by an *insertion sequence*. Speakers insert an additional sequence before the original adjacency pair is completed. Together, the adjacency pair and insertion sequence form an exchange kind of structure, as illustrated by (2):

(2) a. **John:** Can I meet with you next week?
b. **Diana:** Where would you like us to meet during this lockdown?
c. **John:** The park is open.
d. **Diana:** All right!

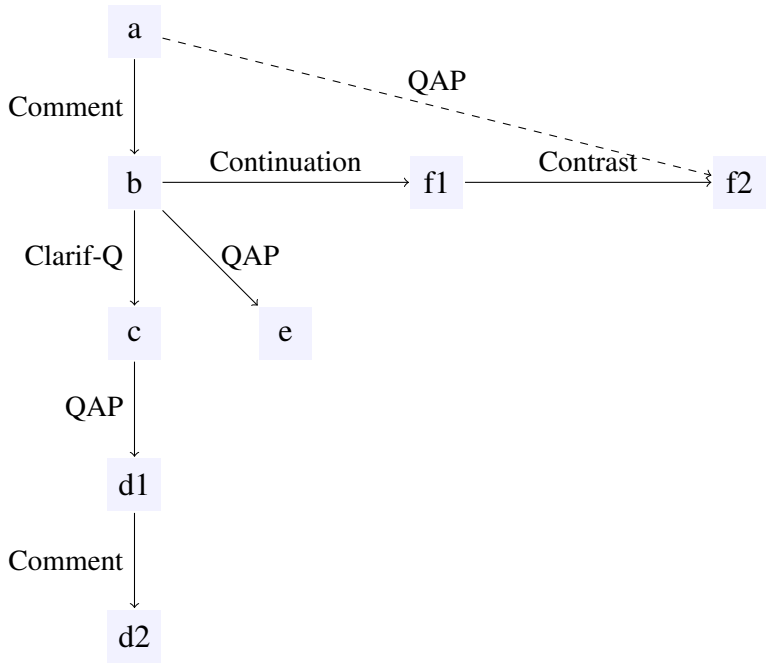
In example (2), Diana in (2-b) is requesting clarification of (2-a) by questioning where they could meet during the current lockdown (she doesn't seem to mind meeting John, but doesn't understand where they could meet). John answers in (2-c), and then Diana accepts by answering the first question, (2-a).

An insertion sequence can be long, leading to very distant attachments, as illustrated in example (3):

(3) a. **John:** Can we get out next Tuesday?
b. **Diana:** Do you remember the first time we went out?
c. **John:** That was when the flamenco festival was held in the city?
d. **Diana:** [You wore a pirate costume,]$_{d1}$ [nothing to do with the festivities' theme.]$_{d2}$
e. **John:** On the Garonne River wharf in late July back in 2010.
f. **Diana:** [Well, there won't be any festivities on Tuesday,]$_{f1}$ [but I am looking forward to seeing you.]$_{f2}$

We naturally succeed in inferring the discourse relations between the segments of this dialogue using the same as those in STAC corpus: a Comment between (3-a) and (3-b), a Clarification-question between (3-b) and (3-c), a Question-answer-pair between (3-c) and

(3-d1), another Question-answer-pair between (3-b) and (3-e), a Continuation between (3-b) and (3-f1) (the structure formed by EDUs b, c, d1, d2 and e can be viewed as a CDU), a Contrast between the two EDUs of (3-f), finally a Question-answer-pair between (3-a) and (3-f2). There are six discourse units between the two clauses that provide arguments to this last relation. All these attachments can be represented with the following SDRS for better clarity (the longest attachment is dashed) :



These long-distance attachments, which often provide discourse-central information, are very difficult to infer computationally because they are scarce, and some approaches simply do not take these long-range relations into account (as we have seen in Chapter 2 for the RST framework of Mann and Thompson, 1988). Through this little example, we can see the importance of following what happens at each moment of the conversation, to understand if people are still talking about the same subject, or if they have been sidetracked, or have moved on to another point. This example (3) was made up, but this phenomenon is very common in the STAC corpus and can be present in any multi-party conversation.

Overlapping/Separate Threads

Another feature of dialogues and multi-party conversations is that they can give rise to *overlapping threads*. Overlapping threads are created when two or more participants engage in two or more connected (via discourse relations) exchanges that are simultaneously and independently developed (in relation to the topic). Figure 3.7 (from Asher et al., 2020)

illustrates three discourse threads, represented by a solid, dashed and dotted lines. (Note that these lines do not reflect discourse relations, click here for full graph visualization with relations).

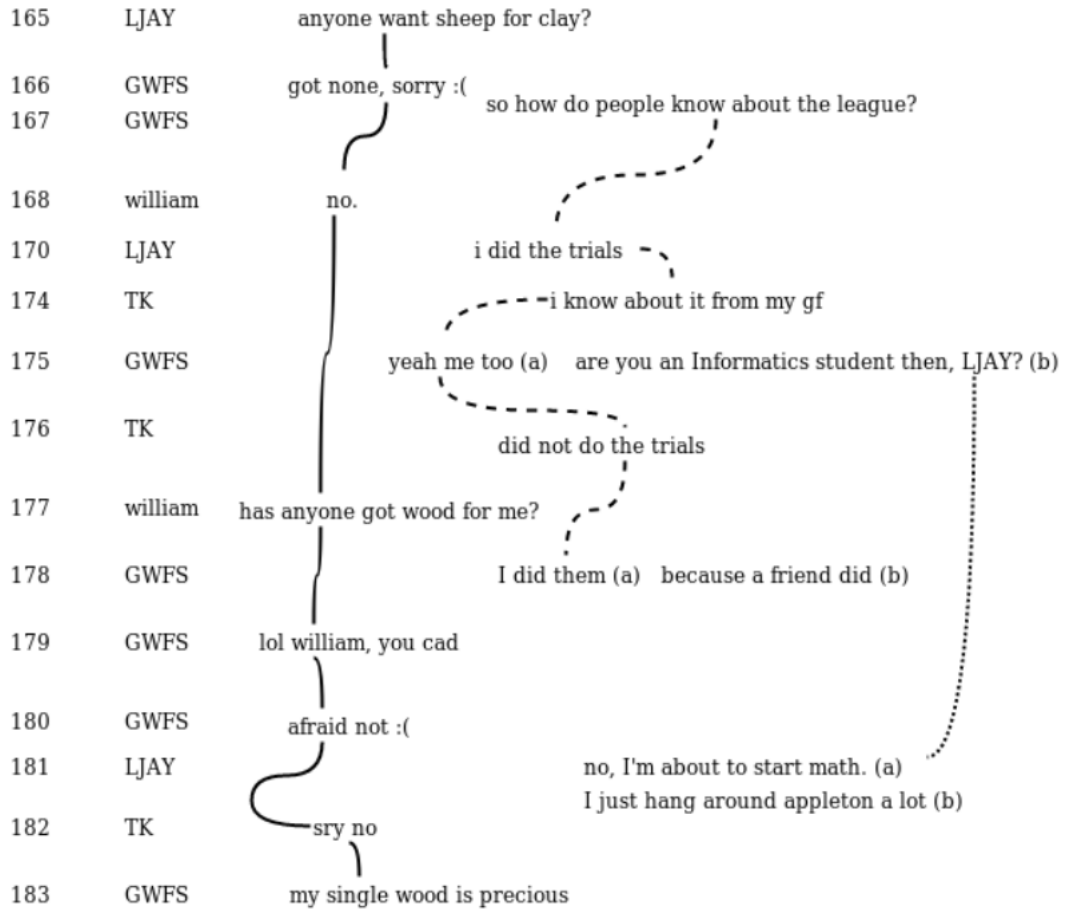


Figure 3.7 Three threads.

Intuitively, these strands do not have much to do with each other and so we have a forest rather than a tree-like structure. A tree-like structure can be recovered if we consider the core structure in the dialogue representation, the backbone, and then attach the separate threads as branches to this backbone. These sidetrackings often occur when, in the course of a meeting or any conversation settings, someone brings up a topic that is partly related, but does not actually refer to the question at hand. Phrases such as *By the way,..., Incidentally, Before I forget,..., While we're on the subject,..., On the subject of (noun),...* or *Out of interest,..*, are often used when someone sidetracks and introduces a new, but related, subject. Having got sidetracked, participants can get back on track and return to the main subject.

Much work has been carried out to identify and analyze threads in different discussion platforms or email exchanges (Aumayr et al., 2011; Carenini et al., 2007; Elsner and Charniak, 2010, 2011; Joty et al., 2013b; Kim et al., 2010; QIU and JIANG, 2013; Wang et al., 2011a, 2010, 2011b). Several types of cues have been used such as the *reply-to* option or the mention of the person being addressed with the common symbol @ used in instant messaging platforms.

Addressees

Phenomena such as truly non-tree-like structures and overlapping threads arise mainly because several agents are involved in the discussion. Knowing who is addressing whom can help to infer the complicated attachments involved in such examples, but how do we know who is being addressed when there are more than two participants?

Unless an addressee is explicitly mentioned, the task of inferring the addressee can be rather complicated. This sometimes depends on the knowledge we have of the participants, for example, if we ask a question about a specific result of a performance achieved and known only by person A, we must deduce that the question is directed to person A. In meetings, we often look at the person(s) we are talking to, so visual clues can help to identify the audience being addressed. However, in the context of our project, we will only study written conversations, so such clues cannot help us. This study is often referred to as *addressee identification* or *allocating turns* (Hayashi, 2012; Sacks, 2004).

In the STAC corpus, we have three types of addressee annotations for each DU; *All*, if a DU is addressed to all the participants, *a set of participant pseudonyms*, if it is aimed at one or several players in particular, and *?* if it cannot be determined. The pseudonym of the person addressed is sometimes included in the EDU text (the second EDU (b) of GWFS's turn 175 in Figure 3.7, addresses explicitly LJAY who responded in turn 181), but this only happens when it is *one* person addressed, and often when the intervention is far from the action or DU to which one wants to respond or react. Consider the following example (4) from STAC (Click here for full graph):

(4) 29 **rennoc1:** so anyone else played this before ?
...
32 **Dave:** is it still your turn rennoc ?

After a few players have responded to *rennoc1*, *Dave* returns back to ask if it's *rennoc*'s turn to play. It should be noted that occasionally the names are abbreviated (william by will, or LJay by LJ for example) and are often placed at the beginning or the end of the EDU.

(5) 727 **Dave:** need wheat ?

...

731 **Dave:** rennoc ?

732 **rennoc:** i'm ok for now

In (5), from the STAC corpus, after some players answered Dave's question, Dave indirectly repeats his question by addressing rennoc, who has not yet answered.

Situated Context

Dialogues often take place in some sort of non-linguistic environment. And this environment contains information, i.e. reference to what we see, what we know, what happens, that impacts our understanding of conversational interactions. The meaning of a dialogue depends not only on linguistic content but also on the environmental information that may make the linguistic content evolve over time. Sometimes, ignoring non-linguistic information can introduce gaps in the linguistic content. For example, if person X asks person Y: *Explain to me the economic development of our company.*, and immediately afterwards the same person X says: *Oh great. We've done well this year*. What was missing in this discussion is that person Y may have presented figures that show the company's turnover evolution for example. In one way or another, something has happened that responds to the content of the interactions without being mentioned.

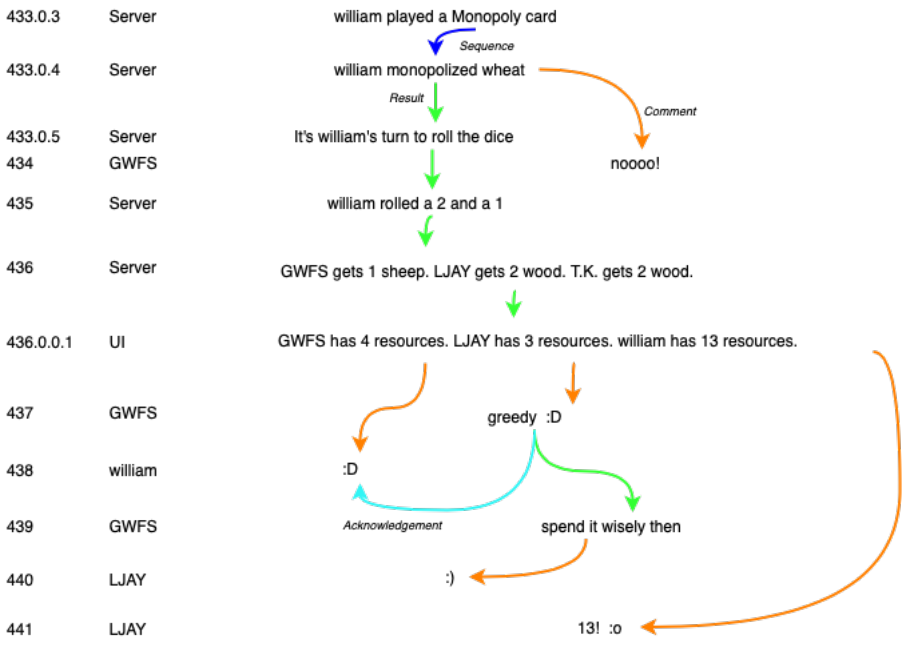


Figure 3.8 Discourse relation between *non-linguistic* segments and *linguistic units*.

As mentioned previously, the STAC corpus was annotated in two versions. The second version is the *situated* one, where non-linguistic units (User Interface (UI) and Server logs) were added to the structure in order to situate the linguistic content (Asher et al., 2016; Hunter et al., 2015, 2018). In Figure 3.8 for example (excerpt from STAC), we can see how the *non-linguistic* context can influence in a relevant way the content and structure of the discourse. The linguistic moves (434, 437, 438, 441) only make sense if they are interpreted in light of the extra-linguistic context. The attachments between *linguistic* and *non-linguistic* DUs shows how the players react to the game moves, but also how the game moves are a reaction to previous game events (players decisions) and discussions. Finding the connections between non-linguistic and linguistic DUs is not a straightforward task. Firstly, EEUs can be grouped together to form CDUs, which means that EEUs are not simply ordered linearly by the time at which their associated events occur. Secondly, the sequential nature of the game events is often interrupted by linguistic DUs, leading to the formation of non-tree-like structures.

Other Phenomena in Spontaneous Text Conversations

Similar to self-repair (Schegloff et al., 1977), in STAC we often find examples of players correcting their own chat moves, corrections made by the same person on their previous intervention. Most of the time when someone is repairing his own utterance, she or he uses the asterisk before the corrected word or expression; a star shaped symbol (*) often used in chat to correct or clarify a spelling mistake when the mistake cannot be edited, as in example (6).

(6) a. **A:** Java layout always feels like a nig
b. **A:** *nightmare

But other corrections exist besides mistypings/misspellings, as illustrated in (7).

(7) a. **A:** oh no
b. **A:** well, that wasn't too bad

The latter type of corrections are difficult to detect, especially if the attachment is of long distance, therefore it is hard to recover a continuous line of thought from chat.

Interjections and emoticons appear frequently in written digital interactions. Most of the time, they serve to moderate the linguistic statements they accompany (*"Good luck all :D"*, *"Thanks :)"*). But sometimes, there are interactions that mainly use these emojis or interjections as illustrated with Example (8):

(8) a. **A:** :D
b. **B:** :O
c. **B:** Wow
d. **A:** I just nicked your wheat :P

We observe that people often write down the *disfluencies* they would have had/said if they had conveyed their message orally (as in Example (9)).

(9) a. **Tom:** When was the last time you cried?
b. **Eva:** Euh. I euh. I
c. **Tom:** It's fine if you don't want to respond
d. **Eva:** Yeah. hhhh :)

If through these examples, we see that there is a meaning to each of these symbols, in STAC the annotations of the links to these DUs (containing these emojis) are often labeled as Comment relations. These phenomena have also been studied separately for conversation analysis such as emojis, laughter, concurrent feedback (*ok*, *uh-huh*, *yeah*) or comprehension checks (Ginzburg, 2016) that also contribute to the establishment of a mutual understanding between participants of the conversation.

The STAC data differ considerably from well written text in other corpora. In addition to the phenomena explained above, the STAC data feature various types of ellipses and/or non-sentential utterances (NSUs) (Fernández et al., 2007; Schlangen, 2003, 2005). Short answers such as the one illustrated in Example (10), or VP ellipsis illustrated in example (11) from the STAC data, are very present traits in our corpus.

(10) **Dave** what have you got to trade ?
Tomm I have got sheep to trade.

(11) I can $_{trade}$ wheat for clay

Talking to more than one people is not as straightforward as we think. During a conversation between two or more people, the linguistic moves people make – greetings, explanations, question-answering, clarifying, contradicting, etc. – are formulated according to the particular history of the exchange and extra-linguistic eventualities at that point. In this section, we presented some problems of inferring attachments in the type of data we want to study: spontaneous multi-party chat dialogues that contain messy text with ubiquitous misspellings, missing punctuation, and other features such as ellipsis or emojis. The diamond structures, long distance attachments, overlapping threads are all very challenging issues for attachment task. However, the semantic conceptualization of non-linguistic events in relation to one

another and in relation to linguistic moves offers us a richer, but also more complex structure to induce.

3.2 Data driven Model Building and Discourse Parsing

Research on linguistic corpora usually focuses on particular phenomena of language and requires a corpus representing these phenomena. When developing a model for a Natural Language Processing (NLP) task, common issues to consider are: the type of input data for each prediction, how to decide on the possible output labels (the approach itself) and how to measure the effectiveness of the model. Annotation campaigns (creation of language resources) are often required prior to performing NLP tasks, not only to train tools, but also to create a baseline for evaluation.

When the task is simplified or straightforward, there are programs and tools that can be used to extract particular items such as online ratings of products, in order to recommend an appropriate list of items that may match a user's preference or interest. But for more complex tasks such as discourse analysis, the need for expert annotators is required, and hand-labeled training sets are expensive and time-consuming to create—-taking months or years for large benchmark sets.

We want to develop a model to automatically build the discourse attachment structures of multi-party dialogues. In our industrial context, data are scarce, task-specific, and noisy. In the following, we will present the computational approaches used for discourse analysis, we will show their limitations, as well as the solutions we want to consider.

3.2.1 Approaches

Discourse parsing is the task of identifying the hierarchical structure of discourse relations in a given discourse or dialogue. For most discourse parsers, regardless of the adopted theoretical framework, building discourse structures computationally requires solving two main problems; first *discourse segmentation* into EDUs, and then *discourse parsing*. In SDRT the *discourse parsing* step can be divided into two stages; the *attachment problem*, which is finding to which DUs in the constructed SDRS an EDU is attached as an argument of a discourse relation, and the *labeling problem*, which involves labeling discourse attachments with labels for discourse relations. In RST analysis, the *parsing* step is generally divided into three sub-tasks; first the *attachment structure prediction* that identify how the different segments are connected/related, then the *nuclearity detection* that identify what is the nucleus of each pair of related segments, and finally *relation labeling* that labels with

RST relations pairs of adjacent text spans. We focus in our work on the *discourse attachment* task following SDRT. Discourse attachment is a difficult problem for automatic processing because attachments are theoretically possible between any two DUs in a dialogue or text, and often graphs include long-distance relations. Extraction of discourse structures for dialogues with multiple interlocutors provides useful semantic information to the *downstream* models used, for example, in the production of intelligent meeting managers or the analysis of user interactions in on-line fora. Applications using these discourse features are becoming more and more numerous, such as automatic meeting summarization, sentiment analysis, or text categorization (Feng et al., 2020; Huber and Carenini, 2020a; Ji and Smith, 2017; Liu et al., 2019; Meyer and Webber, 2013; Nejat et al., 2017).

However, despite considerable efforts to retrieve discourse structures automatically (Duverle and Prendinger, 2009; Fisher and Roark, 2007; Ji and Eisenstein, 2014; Joty et al., 2013a; Li et al., 2014; Surdeanu et al., 2015; Yoshida et al., 2014), we are still a long way from usable discourse models, especially for dialogue. Evaluation of results in discourse parsing has proven complicated (Carlson et al., 2003; Ferracane et al., 2019; Morey et al., 2017). Scores on deep discourse parsing for well-studied and homogeneous resources such as the English RST Discourse Treebank, are still well behind human annotators (Morey et al., 2017). Standard supervised models struggle to capture the sparse attachments, even when relatively large annotated corpora are available. In addition, the annotation process is time-consuming and often fraught with errors and disagreements, even among expert annotators. This motivated us to explore the data programming approach that exploits expert linguistic knowledge in a more compact and consistent rule-based form.

The study of discourse structure in frameworks such as RST and SDRT, has led in recent years to a considerable effort in the creation of corpora (as presented in section 3.1.1), but also the development of discourse parsing models. Several discourse parsers have been proposed (Afantenos et al., 2015; Braud et al., 2017a; Denis and Muller, 2011; Guz and Carenini, 2020; Ji and Eisenstein, 2014; Joty et al., 2015; Matthiessen and Teruya, 2015; Maziero et al., 2015; Perret et al., 2016; Shi and Huang, 2019; Surdeanu et al., 2015; Xue et al., 2016; Zeldes, 2017).

Discourse parsing algorithms following the RST framework have been proposed; using statistical, rule-based, greedy bottom-up or transition-based methods (Feng and Hirst, 2014a,b; Hernault et al., 2010; Joty et al., 2015; Polanyi et al., 2004; Reitter, 2003b; Soricut and Marcu, 2003; Subba and Di Eugenio, 2007). Rule-based approaches, that manipulate syntactic and lexical information, require the creation of a large number of rules and have proven infeasible up to now because of the heterogeneity of all possible texts. Other algorithms use human-annotated data sets to train their models. In addition, constituency-based

RST does not allow non-adjacent relations and is therefore not adapted to the analysis of dialogues we want to achieve.

Previous work on discourse structure prediction following SDRT has given fairly good results with basic discourse structure models. (Muller et al., 2012) is the first paper we know of that focuses on the discourse parsing attachment problem, albeit for monologue. It targeted a restricted version of an SDRT graph and trained a simple MaxEnt algorithm to produce probability distributions over pairs of EDUs, what we call a *local model* with a positive F1 attachment score of 0.635. They further applied global decoding constraints, using A* algorithm, to produce a slight improvement in attachment scores. (Afantenos et al., 2015) used a similar strategy for dialogue on an early version of the STAC corpus. They first applied a MaxEnt model to each pair of EDUs to form a local dependency structure for each dialogue from STAC. The model was trained on a SDRT structure that was transformed into a dependency graph by replacing every attachment to a CDU with an attachment to the CDU's head. In the training set, each pair of EDUs was identified by a set of features grouped into three categories: Positional (such as the position of each EDU in the dialogue), Lexical (whether the EDUs end with a question mark), and Parsing (e.g., the dialogue acts). They then applied the Maximum Spanning Trees algorithm to construct the global structure. (Perret et al., 2016) targeted a more elaborate approximation of SDRT graphs using MaxEnt model on the same version of the STAC corpus and reported a local model F1 attachment of 0.483. They then used Integer Linear Programming (ILP) to encode global decoding constraints particular to SDRT to improve the F1 attachment score to 0.689. However, because these approaches require two separate stages[6], they do not use efficiently the local information of the EDU pairs in the decoding models, and also the local dependency prediction cannot take advantage of the *global* predicted structure for better dependency parsing. Later in this dissertation, we will compare our modeling approach to these three previous papers which used the STAC corpus. Recent approaches try to re-establish this link between local content and the global structure using an incremental approach (Shi and Huang, 2019) to build the structures jointly and alternately. Shi and Huang (2019) reported 73.2 F1 score, the best known score on the prediction of global attachment structure on the early version of the STAC corpus.

Current discourse structure solutions are based on supervised machine learning that suffer from a lack of annotated data sets. In recent years, machine learning approaches have made enormous progress due to the advent of deep learning (DL) models. This progress failed to benefit discourse analysis, because DL models are massively more complex than

[6]These discourse dependency parsing approaches adopted a pipeline framework by first estimating the *local* probability of the dependency attachment between all pairs of EDUs, and then by constructing a complete discourse structure using decoding algorithms.

most traditional models and therefore require a proportionally larger amount of labeled training data. How to deal with these data issues for our task of discourse parsing attachment prediction?

3.2.2 The Importance of Data Sets

The main reason for the limited success of discourse parsers mentioned above is that they require large volumes of *expertly* annotated data to train their models, data that is very costly and time-consuming to create, and domain specific. The data sets that have been constructed have required enormous effort and time, and even so aren't often enough to handle difficult questions about discourse parsing. In addition these data sets are almost impossible practically to extend and often their consistency is an issue. The data sparsity problem applies in particular to neural methods; due to the limited size of labeled data in existing corpora, it is quite hard to train a data-hungry neural model without any prior knowledge.

Moreover, even a single expert annotator, on a single data set, has difficulty annotating data in a consistent manner (Marcu, 1997, 1998; Rath et al., 1961). Complex semantic tasks like discourse annotation are thus sensitive to many factors and background beliefs that are difficult to control. It has been shown that incoherent annotations lead to limited performance of the tools trained with them (Alex et al., 2006; Reidsma and Carletta, 2008); unreliable systems, generated responses are meaningless or inappropriate in conversational systems as we showed through some examples in Chapter 1.2. The ambiguity of annotations often depends on the particular theoretical framework applied (Ferracane et al., 2019). As we explained in Chapter 2, RST can construct several final discourse structures from the same text, and this is due to the ambiguity of the definition of the RST discourse relations that relate to the intentions that the annotator assumes the author or the reader has. The annotation of a discourse following SDRT can also produce different structures, but its logical framework ensures that we can rigorously check for consistency and coherence.

The use of some form of weak supervision

To avoid asking experts to create costly annotated training data, and to satisfy industrial demand for inexpensive learning, research in machine learning developed various approaches to learn from a limited number of examples with supervised information. These techniques are grouped in a machine learning paradigm called *Few-shot Learning* (FLS) (Wang et al., 2020). We often hear about active learning (Druck et al., 2009), semi-supervised learning (Chapelle et al., 2009), or multitask learning (Augenstein et al., 2015; Caruana, 1998)

settings. These are usually used when we already have labeled samples (even from another domain/task). In our context, we want to annotate *unlabeled training data*, data that do not have a label for every training example that indicates its ground-truth output. In this setting where prior knowledge is unlabeled data, FSL becomes weakly supervised learning problem (Zhou, 2018). Instead of having a ground-truth labeled training set we have unlabeled data and weak supervision sources, such that each one has a coverage set and an accuracy (defined as the expected probability of the true label over its coverage set). Weak supervision sources could be various types of higher-level, or otherwise less precise, forms of supervision, which would be faster and easier to provide. The idea is to specify a set of heuristics or other resources, that–if handled properly–could effectively replace thousands of training labels.

Luckily, recent approaches exploit weak supervision sources to infer discourse structures (Huber and Carenini, 2019, 2020b; Liu and Lapata, 2018). Liu and Lapata (2018) have build text document representations hierarchically, by first learning sentence representation (as a sequence of words) using Bidirectional LSTM and a structured attention mechanism with a variant of Kirchhoff's Matrix-Tree Theorem (Koo et al., 2007), and analogously, applied the same operations in the document level, which was viewed as a sequence of sentence vectors. The idea behind the attention mechanism used by Liu and Lapata (2018) is to capture the pair-wise interaction between text units (words, or sentences), to generate a context representation for each unit with *weak structural information*, which form a non-projective dependency tree. The produced trees are the result of Chu-Liu-Edmonds algorithm (Chu, 1965; Edmonds, 1967) to extract the maximum spanning tree from the attention scores. If this approach has not been exposed to labeled training data, this has limited its performance on discourse representations, probably due to the lack of discourse information. Recent work used distant supervision approaches from sentiment-annotated data sets, to generate RST discourse structures (Huber and Carenini, 2019, 2020b). They combined a variant of multiple-instance learning (MILNet) (Angelidis and Lapata, 2018) and an attention mechanism to compute sentiment values and attention scores (that encode the nuclearity/importance attributes) on the document-level and the segments-level, that are combined to run an optimal CKY-style tree generation algorithm (Jurafsky and Martin, 2014). To scale for arbitrary length document restricted by the CKY approach, they applied heuristic beam-search strategies that enhance the discourse tree structure prediction (Huber and Carenini, 2020b). These approaches allow existing data to be extended with discourse trees which can be used as training data sets for neural approaches. The use of distant supervision (along with other task-guided annotations for the construction of discourse trees), or the use of weak structural constrains on the data without external parser, allowed the structures produced to be generalized to mitigate domain

and task specificity (Ferracane et al., 2019; Huber and Carenini, 2020b; Liu and Lapata, 2018).

Weak supervision approaches for NLP tasks, where we either don't have the labels, or it is too expensive to gather them, are what we want to exploit in this thesis. We want to use the insights that can be gleaned from part of the data we want to label (all what has been described in Section 3.1 of this chapter for the STAC corpus), but also the constraints specific to the theory we chose to follow (SDRT as described in Chapter 2) as weak supervision sources that need to be effectively combined to refine attachment structures in a data-driven manner.

3.3 Conclusion

Spoken discourse tends to be less premeditated than written discourse and it is prone to hesitations, corrections and other disfluencies inherent to oral communication. The ubiquity of instant messaging applications have given rise to a new mode of written communication that is extemporaneous, in virtue of the high rate of interactions and immediate responses. In multi-party conversations, paying attention to what was said, by whom, to whom, how it was said, give us the context of the interactions and the flow of the ideas' organization. Trying to find a logical form of natural language interactions can be very useful for automatic inference dialogue models, especially if these interactions are performed with a machine to access or process symbolic data. Simply restricting the processing of queries and their responses to binary treatments is not enough to capture contextual semantics.

We have shown in the Section 3.1.3 that spontaneous multi-party conversations in text format are characterized by a number of specific features. Certain data formats provide us with important information for the analysis of interactions such as speaker identification, to which utterance one comments and even addressee tags. These elements are meant to assist us in following the threads of discussion in order to understand what is being said. It is also necessary to find a way to identify automatically through the contents of the utterances and the information provided, the function of each statement in the whole conversation and how they are related. The STAC data is, to our knowledge, one of the largest annotated corpora with SDRT available for multi-party dialogue analysis. It was therefore chosen to be our study and evaluation data. Yet it provides too little data for predicting dialogue structures using supervised methods.

The large number of discourse annotation campaigns have not solved the problem of lack of data to effectively train deep learning algorithms. All discourse corpora are limited in domain and in size, since annotation is time-consuming and very complex especially

when domain expertise is required. The sparsity of discourse data makes learning difficult, especially considering that discourse analysis involves several complex and interacting factors, ranging from syntax and semantics to pragmatics. Another difficulty is the need to collect corpora for each new task, to find examples of the patterns we want to analyze.

To circumvent these problems we investigated the possibility of automatically and accurately generating large annotated data sets with the Snorkel's data programming method that uses high-level knowledge in the form of weak labeling sources. We will present the Snorkel framework in the next Chapter 4, explaining first the theoretical bases used in its generative models and the different stages of its pipeline. We will then describe in detail in Chapter 5 how we have used Snorkel and how we have adapted it in order to predict discourse structures in the STAC dialogues using SDRT theory to build our set of weak supervision sources.

Chapter 4

Foundations: Tools for our project

As we saw in Chapter 3, it is difficult to collect large corpora for specific tasks. Moreover, even when large corpora are available, it is very difficult, time-consuming and costly to annotate these data. While the developed discourse parsers obtain fairly good results, they are generally trained on these manual annotations and are very limited in their transferability to other data.

When we turn to our industrial use case, then we therefore place ourselves in the situation where we don't have large and well-written corpora. And as we noted in the previous chapter, building a discourse parser without data isn't an option.

In this chapter we lay the foundations for what people call *weakly supervised learning*. In this set up, we won't assume ground truth annotations that we can learn from. Rather we will estimate these ground truth annotations via a statistical technique exploiting several possibly noisy information sources we observe and/or assume in the unlabeled data we want to analyze.

An approach to estimating ground truth labels with easy algorithms is to represent these information sources in graphical models, in which we can learn or estimate dependencies between these sources as well. We will first explain why and how the use of graphical models facilitates inference in the context of our weak supervision. We will then detail two types of graphs, factor graphs and junction trees, which are used in the model we want to explore. Finally, we will present in detail the Snorkel workflow, an open-source system that uses a weak supervision method to apply labels to large data sets.

4.1 Introduction to Graphical Models

Graphical models have become very important because they facilitate the computation of different algorithms directly on the graph (Nielsen and Jensen, 2009). They cover three

fields; statistics, basic graph theory and computer science. They allow complex systems to be built by combining simpler parts using the conditional probabilities of statistical theories and the properties of graphs. In the following section, we will explain through examples the fundamental bases of graphical model representation. The basic idea is that we are trying to represent the knowledge, or observations, in a graphical form, where nodes are going to correspond to the variables, i.e. the sensors of our model, or the characteristics that could be observed, and the edges are going to represent the statistical dependencies between the variables. We don't know the dependencies of the variables in the data. We can only see the co-occurrence or correlations between those variables. Thus we need to deal with the lack of knowledge of the dependencies using other means.

4.1.1 Representation in Graphical Models

Throughout this section, we will illustrate the reasoning behind the graphic models with the basic example of a disease diagnosis. We could take any other example; an aircraft motion with the different sensors such as the aircraft mass and weight, speed, force, level flight, velocity, location and time. The doctor knows that there are factors that may increase the risk or cause lung cancer. Let's imagine a doctor who can take several observations or diagnoses for each of his/her patients: weight, age, whether the patient has lung cancer or not, temperature, coughing, diabetes, bronchitis, whether the patient smokes or not, etc. The doctor has a collection of information on his/her patients where each patient is described by a vector of Boolean or discrete variables. One possible approach would be to construct a graphical representation that indicates the possibility of a disease based on symptoms. At the beginning, we have our various observations which can be represented by circles (nodes). Then we link these variables according to their correlations (edges).

Why do we need graphical models representation? Some statisticians would say that they don't need graphs, they only need the equations. But graphs have become more and more popular and there are several reasons for this. First, graphs are very intuitive. They are a very nice way of globally visualizing relationships between variables (discourse structure, parsing graphs, neural networks or even simpler such as family trees). Importantly, graphs allow us to abstract out the dependencies between the variables from the details of their parametric forms in equations. Pearl and his colleagues (Geiger et al., 1990; Pearl, 1982, 1986, 2014; Pearl and Verma, 1995) were among the pioneers to argue that uncertain information could be efficiently managed if one takes advantage of conditional independence relations in a graphical representation. The conditional independence relationships between the variables is a very important concept in graphical models that we will explain briefly further down in this section to understand the main usefulness (but this concept becomes difficult to handle

in the presence of hidden common causes in graphs). Thus we can answer questions like "Is A dependent on B given that we know the value of C?" just by looking at the graph and not caring about the distribution formed by dependencies, or if variables A, B and C are binary, real, continuous or discrete. Another really important reason to use graphical models is that they allow us to define general message-passing algorithms on the graph that implement probabilistic inference efficiently. Thus enabling us to answer questions like "What is $p(A|C = c)$?" just by sending messages in the graph, without enumerating all settings of all variables in the model, in other words without computing the joint probability of all variables.

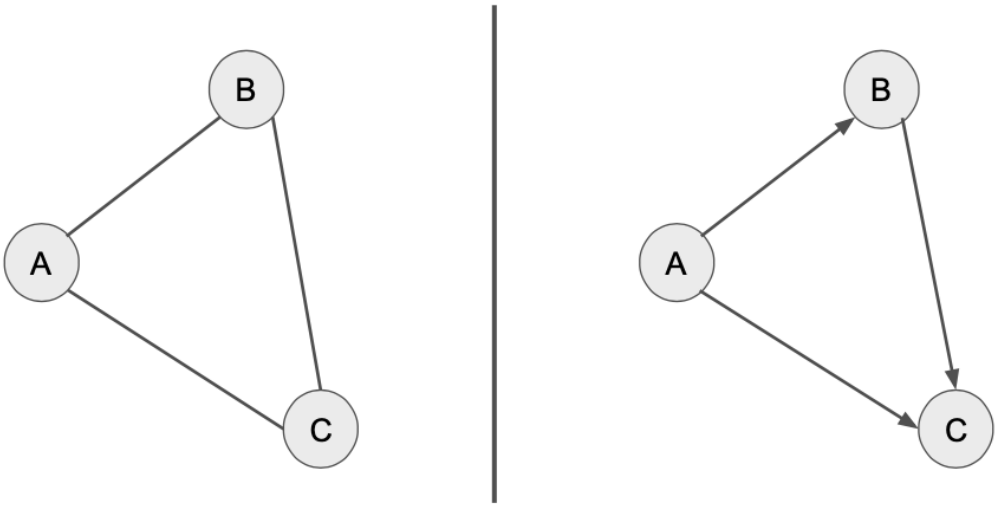


Figure 4.1 A simple example of an undirected graph (left side) and a directed graph (right side).

There exist different types of graphs. A distinction is drawn between **Undirected Graphical Models** (UGM) and **Directed Graphical Models** (DGM). Undirected graphs have edges that do not have a direction. The edges indicate a two-way relationship, in that each edge can be traversed in both directions. The Figure 4.1 shows a simple undirected graph with three nodes and three edges. Directed graphs have edges with direction. The edges indicate a one-way relationship, in that each edge can only be traversed in a single direction. The Figure 4.1 shows a simple directed graph on the right side.

Directed Acyclic Graphical (DAG) model is one of the DGM that is widely used. The concept behind its representation is that we consider arrows pointing at children (descendants) from the parents (ancestors) so as we won't form a cycle (directed graphs without directed cycles). The DAG corresponds to a factorization of the joint probability distribution of the form:

$$p(X_1,...,X_n) = \prod_{i=1}^{n} p(X_i|X_{par(i)}) \tag{4.1}$$

Where $par(i)$ are the parents of node i.

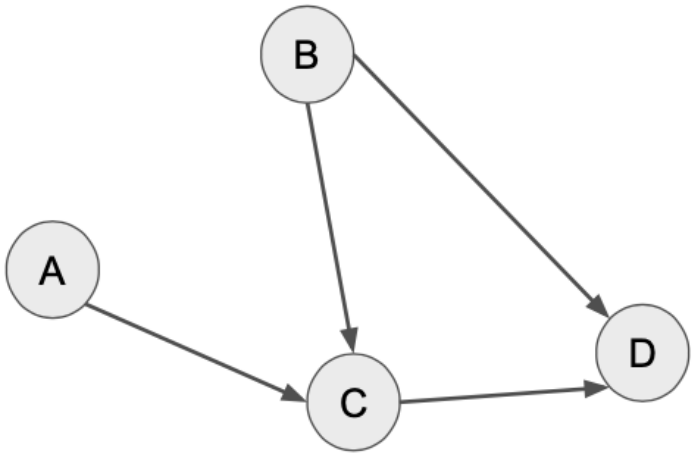


Figure 4.2 A Directed Acyclic Graph (DAG) example. In this graph $A \not\perp\!\!\!\perp D$ but $A \perp\!\!\!\perp D|C$

For example, the joint distribution of the DAG in Figure 4.2 factors in the following way (Equation 4.2):

$$p(A,B,C,D) = p(A)p(B)p(C|A,B)p(D|B,C) \tag{4.2}$$

One of the reason why DAGs are interesting to use, is that they represent efficiently the causal relationships. In Figure 4.2, if the variable A says that the patient gets influenced by people (peer pressure), C says the patient smokes and D says that he has yellow fingers, so it means that A causes C, and C may cause D, so A causes D indirectly through C, following the direction of the arrows, not the other way. D being true does not cause A.

Factoring a probability distribution into a set of smaller conditional probability requires that the variables be conditionally independent. We are interested to know if some variable X is independent of some variable Y given some set of variables V (that dseparates X from Y). This query is simple on DAGs because we can assume that directed edges are causal connections. Formally, we can write $X \perp\!\!\!\perp Y$ which means that X is independent of Y ($p(X,Y) = p(X)p(Y)$), and this implies $p(X|Y) = p(X)$. X is independent of Y conditioned on V, written $X \perp\!\!\!\perp Y|V$, iff $p(X|Y,V) = p(X|V)$. We note that if $X \perp\!\!\!\perp Y|V = Y \perp\!\!\!\perp X|V$ thus it

implies that $p(Y|X,V) = p(Y|V)$. A generalization of this is called Markov Boundary for X. This defines the independence of a variable X relative to all other variables given its parents, its children and its parents of its children (the union). The Markov condition is an assumption made in Bayesian probability theory, that every node in a DAG (a Bayesian network) is conditionally independent of its nondescendants, given its parents. The conditional independence statement has computational implications because we can ignore a part of the graph when computing on a specific set of variables.

Let's give a concrete example to show this difference between conditional independence and marginal independence. Having yellow fingers should be independent whether we are influenced by people or not. A typical situation where there is a conditional independence is that both X and Y result from the same cause, V. For example, smoking causes yellow fingers and lung cancer.

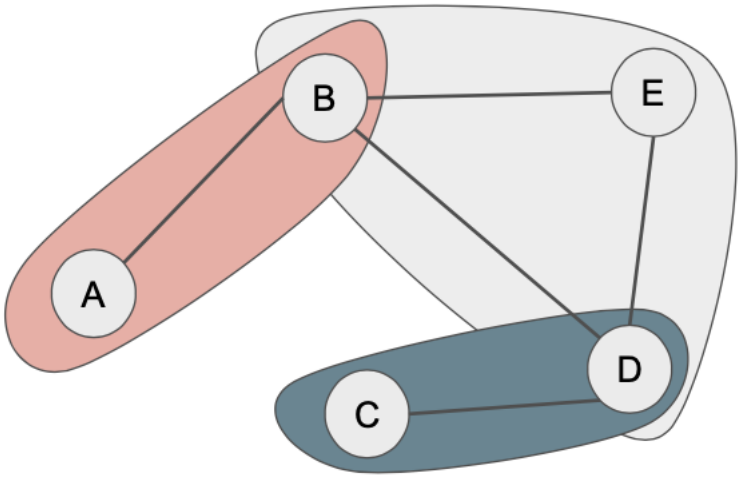


Figure 4.3 An undirected graph with its fully connected sub-graphs.

UGMs are also coming back in popularity for several reasons. Undirected graphs correspond to a particular factorization of the joint distribution that involves looking at all the cliques, i.e. the fully connected sub-graphs. This joint distribution is a product of non-negative functions on the cliques re-normalized. The Equation 4.3 shows the joint distribution of the undirected graph of Figure 4.3, where ϕ is a non-negative function and $1/Z$ is a normalization constant (partition function) to make sure that the product of the non-negative functions, when we sum over all the variables, sums to one (or integrates to one).

$$p(A,B,C,D) = \frac{1}{Z}\phi(A,B)\phi(B,E,D)\phi(C,D) \tag{4.3}$$

Conditional independence for undirected graphs is expressed as follows: every node is independent of all the other nodes given its neighbors. So in Figure 4.3 for example, D is independent of A given B, E and C.

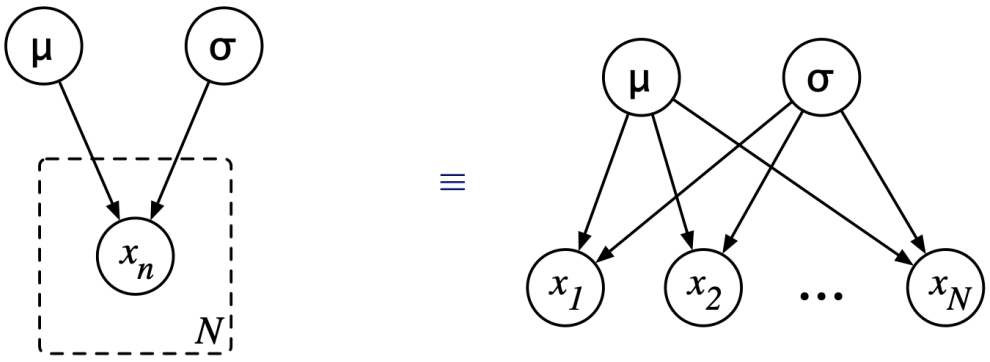


Figure 4.4 Equivalent graphical representations of a statistical model (plate notation on the left shows that the variable x_n gets repeated N times).

A statistical model also corresponds to a particular graphical model. If we consider the following simple model where a data set of N points is generated (or distributed) independently and identically (i.i.d.) from a Gaussian with mean μ and standard deviation σ, this corresponds graphically to the two graphs in Figure 4.4 and to the following joint distribution in Equation 4.4 (we assume that the priors on μ and σ are independent in this simple example):

$$p(x_1,..,x_N,\mu,\sigma) = p(\mu)p(\sigma)\prod_{n=1}^{N} p(x_n|\mu,\sigma) \tag{4.4}$$

Inference in Graphical Models

How do we use graphical models? Inference corresponds to evaluating the probability distribution over some set of variables, given the values of another set of variables. In the graph example in Figure 4.2, if we assume that each variable is binary (that it has only 2 values so that it's Boolean (0,1)), a naive method to compute $p(A|C=c)$ is as follows:

$$
\begin{aligned}
p(A, C = c) &= \sum_{B,D} p(A, B, C = c, D) \\
p(C = c) &= \sum_{A} p(A, C = c) \\
p(A|C = c) &= \frac{p(A, C = c)}{p(C = c)}
\end{aligned}
\tag{4.5}
$$

As the variables are binary, the first sum over variables B and D corresponds to $2x2$, i.e. 4 terms, and we have to compute this sum twice because A has two different values ($4x2 = 8$). When we have $p(A, C = c)$, we can sum up the variable A to get $p(C = c)$ which corresponds to 2 terms. Finally, we have to do two divisions to compute $p(A|C = c)$. In total, we have computed $8 + 2 + 2 = 12$ quantities with this simple method. The question is, can we do that more efficiently? Because we are only interested in the distribution oven A and C. A more efficient method would be to take into account the fact that the joint distribution, induced by our graphical model representation, factors in a certain way. This corresponds to the calculations of the following Equation 4.6:

$$
\begin{aligned}
p(A, C = c) &= \sum_{B,D} p(A)p(B)p(C = c|A, B)p(D|B, C = c) \\
&= \sum_{B} p(A)p(B)p(C = c|A, B) \sum_{D} p(D|B, C = c) \\
&= \sum_{B} p(A)p(B)p(C = c|A, B)
\end{aligned}
\tag{4.6}
$$

We notice that some variables do not appear everywhere in the probability distribution, in particular, the variable D only appears in $p(D|B, C = c)$. So we can bring one of the two sums on the only term where it appears (second line of the Equation 4.6). The sum over some variable is always one. So the sum over D equals one, so it cancels in the equation. We are left with the last expression (last line in the equation), where we are only summing over B and this represents $2x2 = 4$ terms to compute (2 for the sum over B, and additional 2 terms for the two values of A). In total, instead of manipulating 12 terms, only $4 + 2 + 2 = 8$ will be computed (2 other terms for $p(C = c)$, and 2 other terms for $p(A|C = c)$). It is interesting to see in the graph 4.2 that when we are interested in the relationship between A and C, it turns out that the variable D is irrelevant. This gain may not be significant, but in general, for

large graphs, if we use the conditional independence relationships we can get exponential gains in efficiency (for tree structures, singly connected graphs).

4.1.2 Factor graphs

A factor graph (Kschischang et al., 2001) is a graphical model representation that unifies directed and undirected models. It is an undirected bipartite graph with two types of nodes. Round nodes represent variables, and square nodes represent factors. Factors are simply the conditional distributions in the DGMs. There is an edge from each variable to every factor that mentions it. Factor graphs are a representation that makes calculations for inference even easier by propagating messages from variable nodes to factor nodes and vice-versa. Factor graph models convert a joint distribution into a product of functions of a subset of variables as in the Equation 4.7 for the DAG example in Figure 4.2, where Z is a normalization constant. The graphical representation of this joint distribution is showed in Figure 4.5.

$$\begin{aligned} p(A,B,C,D) &= p(A)p(B)p(C|A,B)p(D|B,C) \\ &= \frac{1}{Z} f_1(A) f_2(B) f_3(A,B,C) f_4(B,C,D) \end{aligned} \tag{4.7}$$

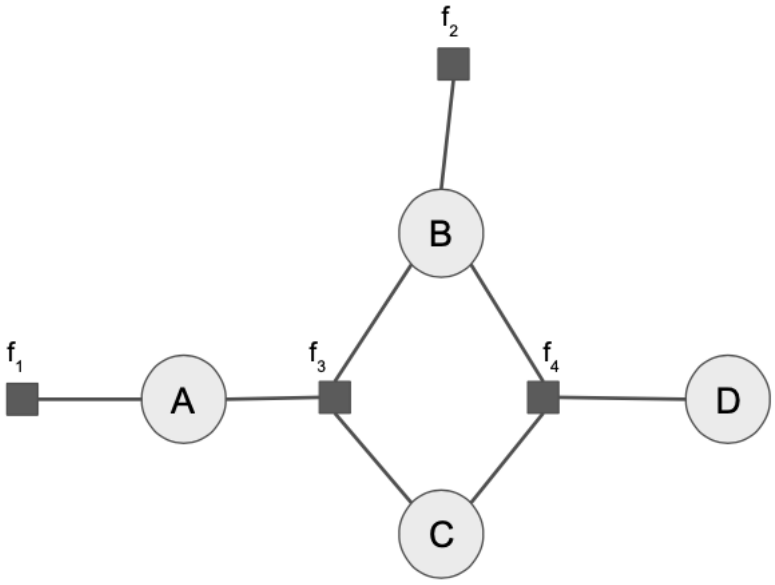


Figure 4.5 An undirected graph with its fully connected sub-graphs.

Let's take a simpler example, which involves 3 discrete, binary variables A, B and C that can take the values {0,1}. A joint distribution over these 3 variables has 7 parameters (we have 8 choices that have to sum up to one). If we only have $f_1(A,B)$ and $f_2(B,C)$ in the factor graph representing this simple example, it means that $A \perp\!\!\!\perp C|B$. So not all of the 7 terms satisfy the conditional independence statement. But what $f_1(A,B)$ could mean? f_1 is a function of two binary variables, so it can be represented by a two by two array of non-negative numbers and they do not have to be summing to one, because the normalization constant Z will take care of that in the distribution. Let's assume we have these random numbers in the f_1 matrix factor $\left(\begin{smallmatrix} 8 & 0 \\ .2 & 4 \end{smallmatrix}\right)$. It is telling that there is a compatibility between A and B for being 0 because of the value 8. The combination of both, A and B being 0 is much more probable than anything else. But $\left(\begin{smallmatrix} 8 & 0 \\ .2 & 4 \end{smallmatrix}\right)$ says that it is logically impossible for A to be 0 and B to be 1, and no matters what the other factors are. While the value 8 in f_1 may get influenced by what is in the factor f_2.

There are two types of factor graph representations ; singly connected (tree representation) and multiply connected factor graphs (with loops in the graph).

Factor Graph Propagation

How do we do inference in factor graphs? We are going to describe a general method for computing on these factor graphs. Computing probabilities in a factor graph is done by propagating messages from variable nodes to factor nodes and vice-versa. The message $m_{x\to f}(x)$ from variable x to factor f is a function of variable x (we can induce the shape of $m_{x\to f}(x)$. If variable x is binary, then $m_{x\to f}(x)$ is a two by one array) and it is obtained by multiplying together for all the factors g that are neighbors of variable x, but not including factor f, all the messages that these other factors g send to x (Equation 4.8 where $n(x)$ denotes the set of factor nodes that are neighbors of x).

$$m_{x\to f}(x) = \prod_{g\in n(x)/\ f} m_{g\to x}(x) \tag{4.8}$$

The message $m_{f\to x}(x)$ from factor f to variable x is also a function of variable x, and it is obtained by summing over all the other variables Y that factor f depends on, not including x, of factor f(Y) times the product of all the messages from other variables h in $n(f)$ to factor f (Equation 4.9 where $n(f)$ denotes the set of variable nodes that are neighbors of factor f):

$$m_{f\to x}(x) = \sum_{Y/\ x} (f(Y) \prod_{h\in n(f)/\ x} m_{h\to f}(h)) \tag{4.9}$$

The rule in factor graph propagation is that if a variable has only one factor as a neighbor, it can initiate message propagation. Once a variable has received all messages from its neighboring factor nodes, one can compute the probability of that variable by multiplying all the messages and normalizing them:

$$p(x) \approx \prod_{g \in n(x)} m_{g \to x}(x) \tag{4.10}$$

When we are conditioning on a variable Z=z, we can change or set a factor that is out of Z, to eliminate values that are not z, when running the message propagation again. If the factor graph is singly connected (tree structure), we only need to propagate messages in two directions (from root to leafs, and then from leafs to root) to have all the marginal probabilities we are interested in.

We notice that all the equations for factor graphs propagation follows the basic sum ($p(x) = \sum_y p(x,y)$) and product rules ($p(x,y) = p(x)p(y|x)$) of probability. The factor graphs propagation is also called the sum/product algorithm. The forward–backward algorithm in hidden Markov models (HMMs), and the Kalman smoothing algorithm in linear Gaussian state-space models (SSMs) are special cases of factor graph propagation (belief propagation). These models can be represented as singly connected DAGs or even in an undirected graphical form.

The probability distribution over variable of interest given the variables that we observed is computed exactly in factor graph propagation for singly connected graphs. For multiply connected graphs (with loops), there are other approximate inference algorithms. The junction tree algorithm solves the exact inference problem for multiply connected graphs, but can be very slow (exponential in the cardinality of the largest clique).

4.1.3 Junction Trees

Junction trees, also called joint-trees, clique trees or tree-clustering, are more easy-to-handle graphs. Why is that? If our graph representation has undirected cycles, local message passing algorithms run the risk of double counting evidence (the problem of common cause) (Beeri et al., 1983). In fact, it can be proved that local propagation is correct if and only if the graph is triangulated, i.e., there are no chordless cycles of length greater than 4. The most common approach is therefore to convert the graph into a tree, by clustering nodes together, to form what is called a junction tree, and then running a local message passing algorithm on this tree. A junction tree is a tree where nodes and edges are labeled with sets of variables. Variable

sets on nodes are called cliques, and variable sets on edges are separators. Cliques contain all adjacent separators, and if two cliques contain variable Y, all cliques and separators on the path between the two cliques contain Y (this last property is called running intersection property). The notion of cliques is closely related to that of complete (sub)graphs (see UGMs factorization in Section 4.1.1).

We can construct junctions trees from the factor graphs, or directly from an UG or DG from the observations (see next Section 4.1.4). The steps are as follows:

- 1) First we need to infer an undirected graph: from a factor graph we only have to make sure that every factor is contained in some maximal clique of the undirected graph. The transformation from DAGs to undirected graph is called **moralization**, where we simply"marry" all parents of each node by adding edges connecting them, then drop all arrows on edges (see Figure 4.6).
- 2) **Triangulation** of undirected graphs: add edges to the graph so that every loop of size > 4 has at least one chord (see Figure 4.7). The triangulation of a graph is not unique and finding the optimal triangulation (in the sense of the minimum number of added edges) is an NP-complete problem.
- 3) Chordal graphs to junction trees by searching for maximum cliques (see Figure 4.8)

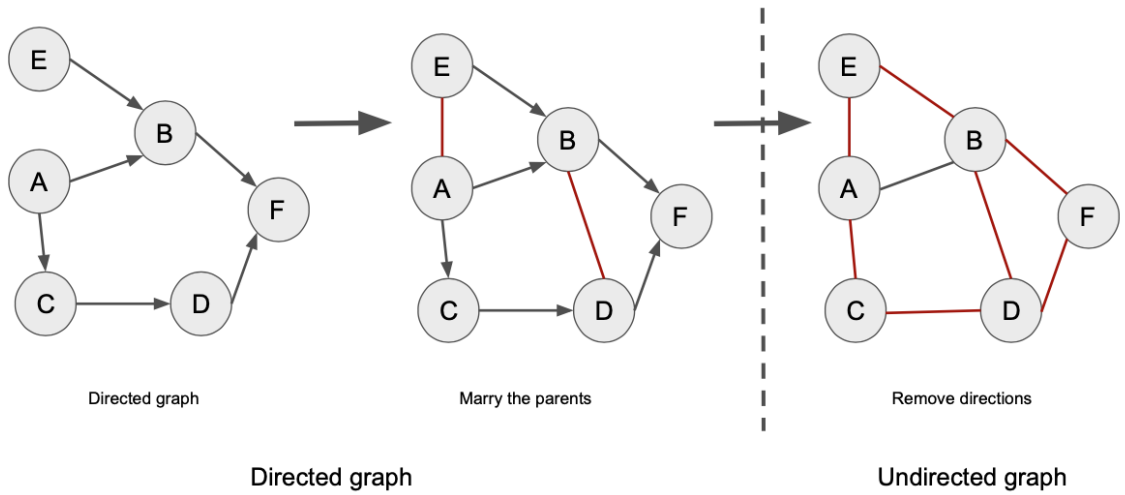


Figure 4.6 Moralization: A DAG (on the left) and its corresponding moral graph (on the right), with newly added arcs shown in red.

Finding a junction tree for a chordal graph is simply a matter of finding a maximum-weight spanning tree, which can be efficiently solved using Kruskal's algorithm or Prim's

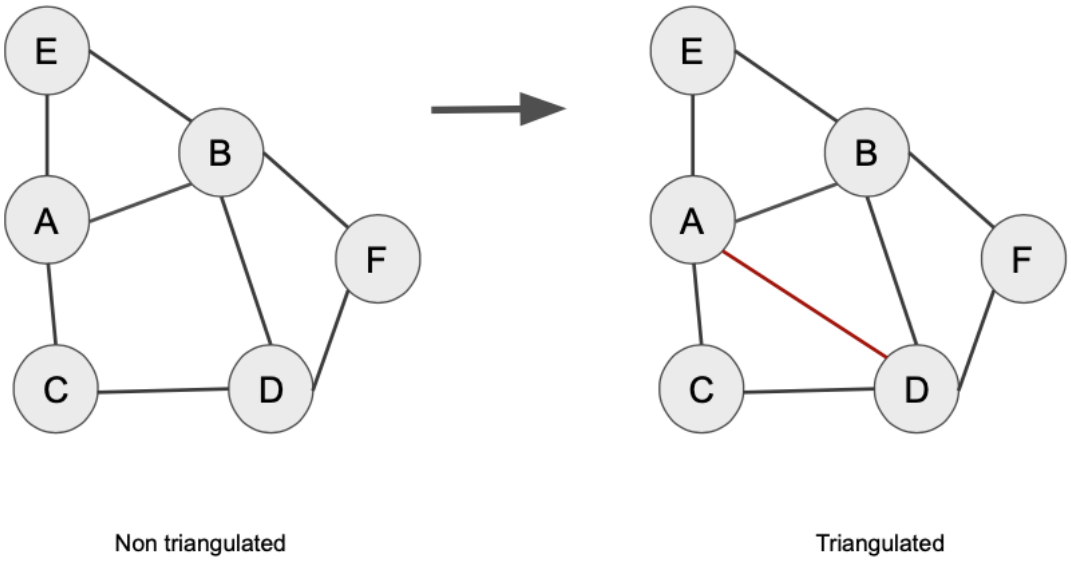


Figure 4.7 Triangulation: Chordal or triangulated undirected graph.

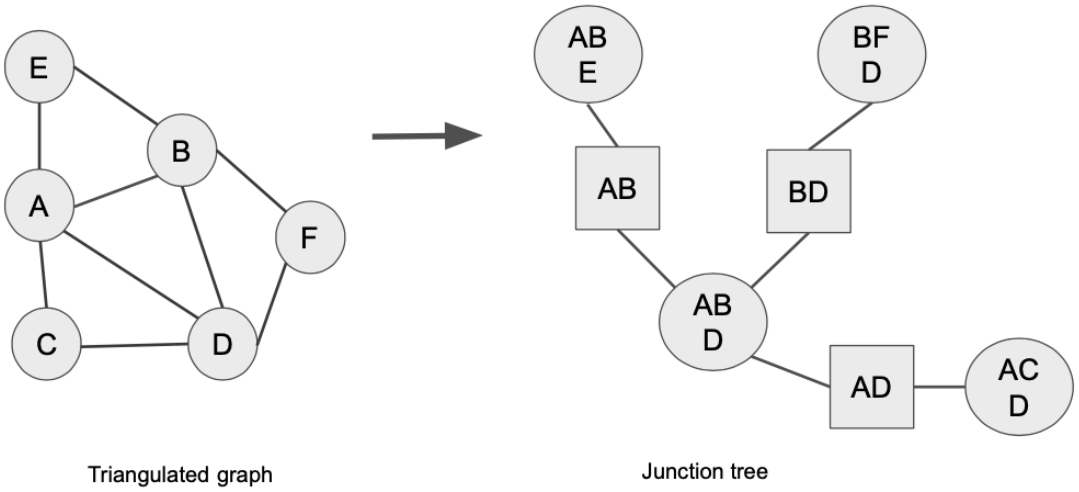


Figure 4.8 Chordal graph to junction tree.

algorithm. For that, we need to find all maximal cliques in the chordal/triangulated graph, and determine all the separators set sizes for the maximal cliques to build a weighted clique graph. If a graph doesn't have a junction tree, the key is to modify it so that we obtain a chordal graph.

Clique-tree Propagation

How do we simplify the joint distribution in junction trees? Inference in Markov networks (UGs) (Lauritzen and Spiegelhalter, 1990) is performed by decomposing the joint distribution into a combination of local potentials that are local joint distributions. The main steps of the junction tree algorithm are:

- Graphical transformations (moralization and triangulation as explained above) that we performed to transform the initial DAG into an undirected junction tree.
- Numerical operations allowing to integrate the initial local distributions into the new structure then perform stabilization operation consisting in propagating marginals in order to guarantee that the marginal distribution relative to a given variable appearing in two adjacent clusters are the same.

In this second step, we incorporate observations by fixing observed variables to take on specific values as a pre-processing step. There are different variants of the Junction Tree algorithm. By accounting for the structure of the edge potentials, we can simplify the Sum-Product message passing equation, resulting in the Shafer-Shenoy algorithm (Shenoy and Shafer, 2008). Additional clever bookkeeping to avoid having to multiply many messages repeatedly yields the Hugun algorithm (Jensen et al., 1990). The computational complexity of this algorithm is exponential in the size of the largest clique making this algorithm efficient only on sparse networks.

Any probability distribution that satisfies the conditional independencies implied by graph G can be factorized as:

$$p(x) = \frac{\prod_{c \in C} p(x_c)}{\prod_{s \in S} p(x_s)} \tag{4.11}$$

where if $s = \{c_1, c_2\}$ (s is a separator in the junction tree) then x_s denotes $x_{c_1 \cap c_2}$.

Having built the junction tree graph, now we need to assign potentials to it. This is easy to do. For edges, the potentials are merely indicator variables which enforce consistency among adjacent cliques. For nodes, the potentials are the maximal clique potentials corresponding to the triangulated graph. The only thing we need to be careful about is that we do not repeat potentials, i.e., we do not assign the same potential in the original graph to multiple maximal cliques in the triangulated graph. This can be done as follows: maintain a list of potentials over all maximal cliques of the triangulated graph. Initialize all the potentials to one. Now go over each potential in the original graph, and assign it to some maximal clique in the

triangulated graph. At the end, the potential for each maximal clique is simply the product of the potentials assigned to it.

4.1.4 Parameters and Structure Learning in Graphical Models

Parameters Learning

When we already have the graphical representation of a model, we have the factorization of the joint distribution over all the variables as explained in the previous section. The conditional probability tables Θ are the parameters of each factor in the distribution. The learning problem is, given the structure of a graph, how do we learn its conditional probability tables from data $D = \{x^{(n)}\}_{n=1}^{N}$ (N examples of the variables x).

For our DAG example in Figure 4.2, we have the following distribution in Equation 4.12. The parameters of this model can be represented as 4 tables: θ_1 has V_A entries (V_A are the values that the variable A, assumed discrete, can take), θ_2 has V_A entries, θ_3 has $V_C x V_A x V_B$ entries and θ_4 has $V_D x V_B x V_C$ entries. The θ_i of variable i can always be represented as a 2-dimensional table ($\prod_{j \in par(i)} V_j$) x V_i.

$$p(A,B,C,D) = p(A)p(B)p(C|A,B)p(D|B,C) \tag{4.12}$$

Let's first describe the maximum likelihood (ML) procedure to learn θ from $D = \{x^{(n)}\}_{n=1}^{N}$. We can also do Bayesian inference over parameters that are unknown, by putting priors on parameters, then we compute the posterior distribution over the parameter given the data, $p(\theta|D) = p(D|\theta)p(\theta)/p(D)$. For Maximum likelihood, we have to take the likelihood function (Equation 4.13 where each term $p(x^n|\theta)$ is represented using the conditional independence structure of the graph) and maximize it with respect to Θ.

$$p(D|\Theta) = \prod_{n=1} N p(x^{(n)}|\theta) \tag{4.13}$$

And then when we take the logarithm of the likelihood 4.14, the log of a product turns into a sum over N data points, and a sum over variables i (for each data point, we have an observation of all variables) of the term $logp(x_i^{(n)}|x_{par(i)}^{(n)})$.

$$logp(D|\Theta) = \sum_{n=1}^{N} \sum_{i} logp(x_i^{(n)}|x_{par(i)}^{(n)}, \theta_i) \tag{4.14}$$

The maximum likelihood algorithm is very simple when the data is fully observed for optimizing the parameters.

In very realistic situations, we often have **hidden variables** (missing observation for some data points). A naive thing to do is that we can throw out the data with variables that are not observed, and calculate the parameters from the other observed data, but this doesn't give us the maximum likelihood solution. This naive procedure is not statistically efficient unless we have a large amount of data observed, which is often not the case. There is a better way to proceed by running the Expectation Maximization (EM) algorithm to optimize the parameters (Dempster et al., 1977). The intuition of EM algorithm is as follows: we have to iterate between applying the E step where we have to fill in the hidden/missing variables by assigning some probabilities, and the M step where we apply complete data learning to filled-in data (estimate the parameters by calculating the probability distribution/density over all possible settings. This can be very hard, except for some simple linear models with Gaussian noise). The goal is to maximize parameter log likelihood given the observable data Y (the log likelihood $p(Y|\theta)$ in Equation 4.15, where we sum over the hidden variables X, of the joint distribution of the hidden and observed variables).

$$log p(Y|\Theta) = log \sum_{X} p(X,Y|\theta) \tag{4.15}$$

The EM algorithm ensures that we have the maximum of the likelihood. It may not be the global maximum, but it is certainly a local maximum of the likelihood. When we have a non-linear model, or a non-Gaussian noise, we can apply sampling methods for the E step such as Markov Chain Monte Carlo (MCMC), Gibbs (Geman and Geman, 1984; Smith, 1991) that originate from the Metropolis-Hastings algorithm (a detailed review of this method is given by Chib and Greenberg (1995)). And we can apply sampling methods in both hidden variables and parameters because EM steps may not be very sensitive when finding a single value of the parameters. A key issue in the successful implementation of any MCMC sampler is the number of runs (steps) until the chain approaches stationarity (the length of the burn-in period).

For easy situation, when we have complete data, Bayesian learning is not more costly than maximum likelihood. But for a situation with incomplete data, it is a little bit harder because we have to infer both parameters and the hidden variables. We can use EM algorithm, or for Bayesian learning other methods like MCMC or Viterbi.

Structure Learning

We assumed previously that we had the structure of the model, i.e. the graph representation. But can we learn the structure of the graphical model given a data set of observations (set of variables)? If we have N variables, and all what we observe is the settings of these N variables; we don't know whether a variable is correlated to another variable. We want to infer the set of edges in the graph model.

There exist two different methods for structure learning. There is a learning approach based on Constraints, where we use statistical tests of conditional and marginal independence and find the set of DAGs whose d-separation relations match the results of conditional independence tests. And there is Score-Based Learning, where we use a global score such as the Bayesian marginal likelihood and find the structures that maximize this score.

For complete discrete observed data D for example, we can do Greedy search algorithm by starting with a graphical model with structure m, and parameters θ, and compute the score of the graph, $score(m)$, from the data set, and evaluate adding or removing an edge to the graph. We can iterate and choose each time the best score. If we have *missing data*, it is more complicated because we have to estimate the scores. In structure learning, there is a lot of research on learning causal relationships in DGMs which we are not going to address in these explanations. In UGMs, the probability of all the variables $p(X|\theta)$ depends on intractable normalization constant $Z(\theta)$, except for tree like structure where we can compute $Z(\theta)$ exactly. There are several solutions to approximate this normalization constant or its derivatives and compute inference for UGMs (Jordan et al., 1998; Murray and Ghahramani, 2012): e.g. Boltzmann machine learning, contrastive divergence, pseudo-likelihood, loopy belief propagation or bounding methods.

Undirected graphs provide a natural description of constraints between k observations/variables X. Mutual compatibilities among variables are described by a factorized joint probability distribution 4.16, where $C_j \subset \{1,...,k\}$ indexes a subset of the variables and ϕ_j is a potential function, parameterized by θ_j, expressing compatibilities among X_{C_j}:

$$p_\theta(X|\theta) = \frac{1}{Z(\theta)} exp\left\{\sum_j \phi_j(X_{C_j}, \theta_j)\right\} \tag{4.16}$$

The partition function $Z(\theta)$ is a sum or integral over all variables configurations:

$$Z(\theta) = \sum_X exp\left\{\sum_j \phi_j(X_{C_j}, \theta_j)\right\} \tag{4.17}$$

The UGM representing the conditional independencies implied by the distribution 4.16 has a node for each variable and an undirected edge connecting every pair of variables (x_i, x_m), if $i, m \in C_j$ for some j. The subsets C_j are cliques of the whole undirected graph. A more general representation of UGM is a *factor graph* with the variables nodes ($i \in \{1, ..., k\}$) and the factors j (edge from variable i to a factor j if $i \in C_j$). Factor graphs are a good representation of the Boltzmann machine that assumes variables X to be binary ($x_i \in \{0, 1\}$), so the undirected graph has edges for all non-zero elements of its weight matrix θ that parameterizes the distribution.

For learning the Boltzmann machine, we usually use a maximum likelihood version of the EM algorithm, where we assume some variables to be hidden x_H and some observed x_O (Ackley et al., 1985). The gradient of the log probability $p(X|\theta)$ is the difference between the expectation under the data distribution $p(X_H|X_O, \theta)$ and the expectation under the distribution $p(X|\theta)$. If we consider a data set of N points that is generated (or distributed) independently and identically (i.i.d.), the gradient of the log likelihood is simply summed over N. Computing expectations of Boltzmann machines is exponential in time. The Gibbs sampler is a popular choice for approximating these expectations (other inference algorithms exist).

Computing the parameters $p(\theta|X)$ using Bayesian inference is of the form of Equation 4.18:

$$p(\theta, X) = \frac{1}{Z} exp \left\{ -\frac{1}{2\sigma^2} \sum_{j<i} \theta_{ij}^2 + \sum_n \sum_{j<i} \theta_{ij} s_i^n s_j^n \right\} \tag{4.18}$$

The prior over the parameters when marginalizing 4.18 is of the form 4.19:

$$p(\theta) = \sum_X p(\theta, X) \propto \mathcal{N}(0, \sigma^2 I) Z(\theta)^N \tag{4.19}$$

This $p(\theta)$ prior is dependent on the size of the data set, and its parametric form is very complicated, favoring weights with large partition functions. To address this intractable problem, the MCMC (Markov Chain Monte Carlo) (Neal, 1993) methods, such as Metropolis or Langevin sampler, are being used to generate correlated samples from probability distribution with unknown normalization. These methods allow to generalize the inference to any Bayesian learning in a general undirected model of the form 4.16. The Gibbs sampler is a popular choice for approximating pairwise marginals for the computation of the joint probability $p(X, \theta)$ gradient or for parameters update (also called Brief sampling). But this algorithm, inspired by Contrastive Divergence work, uses brief sampling starting from the data, X, which gives biased but low variance estimates of the required expectations.

In this section, we talked about an intuitive and computationally useful representation for probabilistic modeling which are graphical models. We have shown the different solutions, more or less simple and accurate, for learning this graphical structure from the observations. The general approach to inference in arbitrary graphical models is to transform these graphical models to ones belonging to an easy-to-handle class of graph representation. In the next section, we will describe Snorkel Ratner et al. (2017a) and will show how it uses this approach to annotate data.

4.2 Data Programming

Weak supervision is an approach to alleviating the issues around the difficulty of creating training data. The problem with manually annotated data is its high cost and lack of flexibility, especially when an expertise is required (see Section 3.1.1). One solution to this concern is to program the data with domain expertise or other weak supervision information. Some examples of sources of weak supervision include domain heuristics (e.g. common patterns, rules, etc.), distant supervision sources (existing resources such as knowledge bases, alternative data sets, or pre-trained models), and crowdsourcing workers. These are known as weak supervision sources because they may be limited in accuracy and coverage.

To help reduce the cost of training set creation, Ratner et al. (2016) introduced the data programming paradigm, along with an open source framework, called Snorkel Ratner et al. (2017a). It uses a weak supervision method to apply labels to large data sets by way of heuristic labeling functions that can access distant, disparate knowledge sources. These labeling functions, that can issue a label or abstain, are represented as variables in graphical models in order to observe their correlations, so as to estimate the labels we are looking for.

4.2.1 Snorkel

The Snorkel project[1] began at Stanford in 2016 with the goal of getting more and more training data by bringing a mathematical and system structure to the messy and often entirely manual process of creating and managing training data. The key idea behind Snorkel, is to take advantage of noisy (or weak) sources of labels, and be able to model their noise using structure learning in graphical models, and combine them so that we can get labels more efficiently. Let's go through the basic steps of this system (Figure 4.9).

After loading in unlabeled data, Snorkel consists of three main steps :

[1] https://www.snorkel.org

- The first step is the labeling rules phase where subject-matter experts (SMEs) users write **Labeling Functions (LFs)** which capture patterns and heuristics or/and connect with external knowledge bases (distant supervision). A labeling function is a Python method which given an input can either output a label or abstain.
- The second part of this Snorkel system is the algorithm that helps to learn and model the previous heuristics that users wrote. This algorithm needs to know how accurate the LFs are, and find out if they are correlated. When having this information, the **generative model** of this step will combine them in an optimal manner to assign training labels to the data that we have. The most important aspect of this step is that it does not use ground-truth data, learning instead from the agreements and disagreements of the LFs that the model represents as variables in a graph in order to use the various structure learning methods presented in section 4.1.4
- Snorkel outputs a set of probabilistic labels which can then be used to train a wide variety of machine learning models (any **discriminative model**).

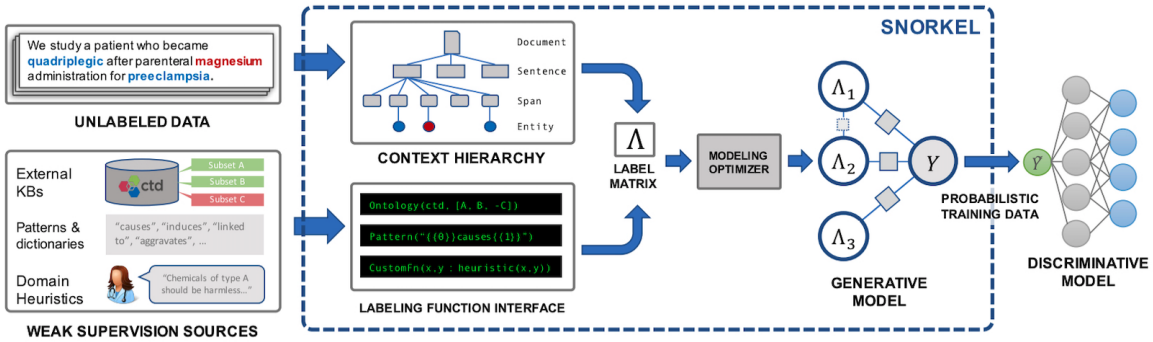


Figure 4.9 An overview of the Snorkel workflow.

We need to access unlabeled data, which should be a fairly easy task depending on the field. Snorkel breaks input data down into a context hierarchy made up of context types (*Context Hierarchy* module in the top left side of the Snorkel system overview 4.9). The set of context types that make sense will be data dependent. For, e.g. binary classifier text-relation extraction, we might extract documents, sentences, spans, and entities. Tuples of relevant entities are then passed to labeling functions as candidates.

4.2.2 Labeling Functions

The first step, which is the most meticulous part of the system, is writing LFs. It is a way of encoding domain knowledge people have about the data they want to label. The LF concept

is quite general. We often use knowledge/expertise/heuristics to form LFs, but it is perfectly adequate to use a general pre-trained classifier as one too. A LF can look for words in a sentence for example, or certain patterns that tell us a good enough signal of whether the data candidate is true label y or false label y. If the pattern is not found the LFs can abstain, i.e. they do not necessarily return a label. The convention in snorkel is to generally output 0 when the LF abstains. We can write multiple of these LFs, but none of them are going to be perfect and cover the whole data. If we could write perfect rules then we wouldn't need to do Machine Learning for our specific classification tasks. So some of the LFs may only assign a label to 10% of the data set, but they can be 90% accurate on that subset. Other LFs may be only 60% accurate but they may have labeled 90% of the data. And these LFs don't have to be mutually exclusive, as the Generative Model will be able to learn their correlations. The choice of polarity and class size is flexible for LFs as well. The important point that characterizes LFs is that they can abstain. We are (then) faced with incomplete observations. One method to improve the estimation of our LFs is to introduce dependencies to estimate the scores according to correlations.

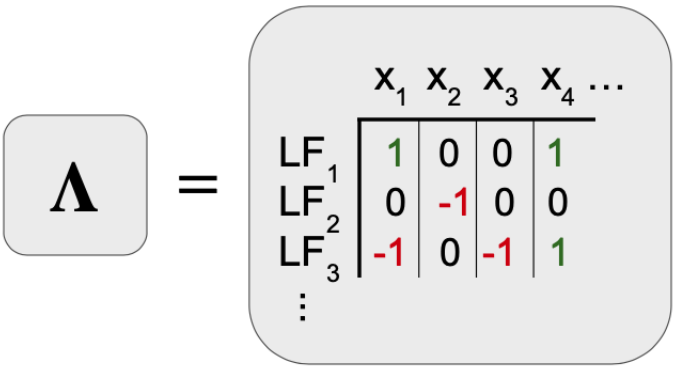


Figure 4.10 Label Matrix Λ obtained by applying the LFs over unlabeled data.

In this Snorkel system, the LFs (that can be any weak supervision sources) can be written with a beginner's knowledge of Python. For simple cases, we can use the built-in declarative LFs instead of writing our own. This was one of the main goal of the Snorkel team; to facilitate data handling and rules composition for everyone.

4.2.3 Modeling Weak Supervision Learning

After writing a collection of LFs, we can apply those on all the unlabeled candidates to create a large number of labels (label matrix Λ in Figure 4.10) that are not perfect, but "good enough" for a potentially huge training data set. We could intuitively use majority voting

to determine the resulting labels. But majority voting works really well only in situations where we don't have many votes on an input candidate (e.g., most of the LFs abstain), and in situations where we have lots of votes. But in-between these two extremes, taking a weighted vote based on modeling LF parameters works better. In order to learn the LFs accuracies, and their correlations, we can not rely on any label because we want to move away from hand labeling the data. Instead the Generative model looks at the overlaps, conflicts and dependencies among the labels assigned by these different LFs to assign probabilistic labels using a graph-based model.; on a factor graph for the first version[2] (Ratner et al., 2017a), and on a junction tree for the second version[3] (Ratner et al., 2019) of the Snorkel system. We are in a situation where there is missing data (since the LFs can refrain from labeling data points), and where we do not know the dependencies between the LFs. It is therefore necessary not only to estimate the missing information, but also to find the structure of these observations. And it is possible to do this through the algorithms applied to graphical models we saw in section 4.1.4. Both versions of the Snorkel generative model are described in detail below.

First version of Snorkel

Let's formally describe the learning problem to get an idea of everything that is taken into account in the calculation of the first version of the Snorkel Generative Model. We will be following the explanations of the three publications (Bach et al., 2017; Ratner et al., 2017a, 2016). The configuration we have of a binary classification task (generalization to the multi-class setting is very simple) is a set of m input candidates x_i and their associated latent random variable $y_i \in \{-1,1\}$ that is their true label (the set of true labels Y). We don't have access to y_i, but we do have n labeling functions $\lambda_1,...,\lambda_n$ whose domain $\{-1,0,1\}$ (for False, Abstain, and True outputs) that can be applied to each x_i to produce $\Lambda_{i1},...,\Lambda_{in}$ outputs. We end up having a matrix, a distribution $\Lambda \in \{-1,0,1\}^{mxn}$ and we want to estimate the joint probability $p(\Lambda,Y)$ where $Y \in \{1,-1\}$ are the predicted labels. The first assumption that Snorkel does is that the LFs are conditionally *independent* given the true label y (the label Y being estimated is dependent on the rules or observations). If we consider a LF λ_j, which gives output Λ_{ij} to input x_i that has true label y_i, we can model the *accuracy* of this LF with ϕ_j^{Acc}. We can set this accuracy function to 1 if λ_j correctly labels a data point, and 0 otherwise:

$$\phi_j^{Acc}(\Lambda_i, y_i) := y_i \Lambda_{ij}$$

[2]https://github.com/snorkel-team/snorkel-extraction
[3]https://github.com/snorkel-team/snorkel

This ϕ_j^{Acc} function is associated to a parameter θ_j^{Acc} which models how accurate each LF j is. The generative model, as specified in 4.20, provides a general distribution of n conditionally independent LFs on m input data points:

$$p_\theta(\Lambda, Y) \propto exp(\sum_{i=1}^{m}\sum_{j=1}^{n} \theta_j^{Acc}\phi_j^{Acc}(\Lambda_i, y_i)) \tag{4.20}$$

We can estimate the parameter θ_j^{Acc} by minimizing the negative log marginal likelihood of an observed matrix $\overline{\Lambda}$ for not giving the correct label (false or abstain outputs) using stochastic gradient descent (Equation 4.21):

$$argmin_\theta - log\sum_{Y} p_\theta(\overline{\Lambda}, Y) \tag{4.21}$$

The gradient is the difference between the sufficient statistic of the joint distribution and the same distribution conditioned on the observations $\overline{\Lambda}$ (4.22); in other words, the gradient for the parameter θ_j^{Acc} is given by the number of examples λ_j correctly labels minus the number of examples it fails to label correctly:

$$\sum_{i=1}^{m}(E_{\lambda, Y\sim\theta}[\phi_j(\Lambda_i, y_i)] - E_{Y\sim\theta|\overline{\Lambda}}[\phi_j(\overline{\Lambda}, y_i)]) \tag{4.22}$$

As mentioned in (Bach et al., 2017), "statistical dependencies arise naturally among weak supervision sources". What are the dependency types modeled in this original Snorkel version? There are four of them (see Section 4 of the article (Ratner et al., 2016) for more details of the computation of these dependencies.):

- dep_similar: when two LFs always agree on a label,
- dep_fixing: when a LF_i fixes the mistakes made by another LF_j,
- dep_reinforcing: when a LF_i reinforces the label outputted by another LF_j ,
- dep_exclusive: when two LFs always disagree on a label.

It is therefore important to model these LFs dependencies with a general model as expressed in 4.23:

$$p_\theta(\Lambda, Y) \propto exp(\sum_{i=1}^{m}\sum_{t\in T}\sum_{s\in S_t} \theta_s^t\phi_s^t(\Lambda_i, y_i)) \tag{4.23}$$

This leads us to model this distribution with a graphical model. The dependencies will be of the form (lf_1, lf_2, type) where T is the set of the dependency types, S_t is a set of

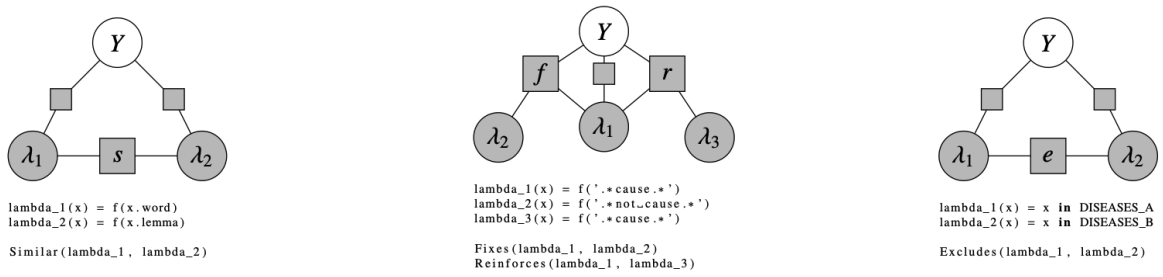


Figure 4.11 LF dependencies examples represented in the factor graph (from (Ratner et al., 2016)).

index tuples of LFs participating in dependency type $t \in T$ (this is similar to the formula 4.16 explained earlier). Snorkel built the generative model, using the dependencies, into a general factor graph of the following form :

$$p_\theta(\Lambda, Y) = Z_\theta^{-1} exp(\theta^T h(\Lambda, Y)) \tag{4.24}$$

Z_θ is the partition function which ensures that p_θ is a distribution and h is a set of factor functions where each h_i represents a factor (we recognize the similarity of these equations with the Equation 4.16, described in section 4.1.4).

We want to learn the parameter θ_s^t. But because of the latent variables Y, we need to estimate the marginal likelihood $p_\theta(\Lambda)$. The solution chosen by the Snorkel team is the Gibbs sampler to estimate gradients (a popular choice as it is the cheapest way to sample). The library *Numbskull*[4], a Python NUMBA-based Gibbs sampler, is used to do a combination of Gibbs sampling and stochastic gradient descent, which is similar to contrastive divergence, for learning the generative label model. This is mainly with *sample_and_sgd* method from the *learning.py* script of *Numbskull* that the Gibbs sampling (sample) and stochastic gradient descent (SGD) are computed. It considers each LF in turn, optimizing the log marginal pseudolikelihood of its outputs conditioned on the outputs of all the other LFs (Equation 4.25 where ε is a hyperparameter that controls the threshold and regularization strength):

$$argmin_\theta - log p_\theta(\overline{\Lambda}_j | \overline{\Lambda}_{\setminus j}) + \varepsilon ||\theta||_1 \tag{4.25}$$

The algorithm in Figure 4.12 details how this is computed. If the gradient is computed in polynomial time (Bach et al., 2017), it is relative to the number of LFs, all possible dependencies, and more particularly to the number of candidates, which can increase the runtime if the data is large enough (we noted this common issue in Section 4.1.4).

[4]https://github.com/HazyResearch/numbskull

Algorithm 1 Structure Learning for Data Programming

Input: Observations $\bar{\Lambda} \in \{-1, 0, 1\}^{m \times n}$, threshold ϵ, distribution p with parameters θ, initial parameters θ^0, step size η, epoch count $\mathcal{T}$, truncation frequency K

$D \leftarrow \emptyset$
for $j = 1$ **to** n **do**
  $\theta \leftarrow \theta^0$
  for $\tau = 1$ **to** $\mathcal{T}$ **do**
    for $i = 1$ **to** m **do**
      for θ_s^t **in** θ **do**
        $\alpha \leftarrow \sum_{\Lambda_{ij}, y_i} p(\Lambda_{ij}, y_i | \bar{\Lambda}_{i \setminus j}) \phi_s^t((\Lambda_{ij}, \bar{\Lambda}_{i \setminus j}), y_i)$
        $\beta \leftarrow \sum_{y_i} p(y_i \mid \bar{\Lambda}_i) \phi_s^t(\bar{\Lambda}_i, y_i)$
        $\theta_s^t \leftarrow \theta_s^t - \eta(\alpha - \beta)$
      if $\tau m + i$ **mod** K **is** 0 **then**
        for θ_s^t **in** θ **where** $\theta_s^t > 0$ **do**
          $\theta_s^t \leftarrow \max\{0, \theta_s^t - K\eta\epsilon\}$
        for θ_s^t **in** θ **where** $\theta_s^t < 0$ **do**
          $\theta_s^t \leftarrow \min\{0, \theta_s^t + K\eta\epsilon\}$
  for θ_s^t **in** θ **where** $j \in s$ **do**
    if $|\theta_s^t| > \epsilon$ **then**
      $D \leftarrow D \cup \{(s, t)\}$
return D

Figure 4.12 Structure learning in Snorkel.

Assuming that there exist some set of parameters which can capture the true dependencies within the model we are using and that all non-zero parameters have at least a minimum magnitude κ, the theoretical guaranty provided in (Bach et al., 2017) to recover the exact dependency structure with a probability of at least $1 - \delta$ is to have m data inputs where:

$$m \geq \frac{32d}{c^2\kappa^2} \log\left(\frac{2nd}{\delta}\right) \tag{4.26}$$

d in 4.26 is the maximum number of possible dependencies a single LF can be involved in, and c is a measure of how much better our dependency estimates are with each LF than without (see Equation (7) in section 4 of the paper (Bach et al., 2017) for the official definition). If we assume that the only dependency types we have are accuracy and correlation dependencies, then we need an input data set of size:

$$m \geq \frac{64d}{c^2\kappa^2} \log\left(\frac{4nd}{\delta}\right) \tag{4.27}$$

Second version of Snorkel

The structure learning in weak supervision setting done in (Bach et al., 2017) depends on the number of data points and not on the graph structure learned on the weak supervision sources Λ. The second version of Snorkel (Ratner et al., 2019) which was released in 2019[5], has a different approach; the learning process depends on the structure of the graph made up of the LFs, and not on the number of data points.

We are in the same problem setup; X is the input data, Y is the true label with (X,Y) drawn i.i.d from some distribution, and we never have access to the label Y; instead we rely on n weak supervision sources that outputs noisy labels λ_i for $0 \leq i \leq n$. The joint distribution of $p(\Lambda,Y)$ is modeled with an undirected graph (as in the previous version from (Bach et al., 2017)) $G = (V,E)$ with $V = \{\lambda_1,...,\lambda_n\}$, and if λ_i is not independent of λ_j conditioned on Y and the other sources, then (λ_i,λ_j) is an edge in G. This dependency graph is then represented in the generative model by its *junction tree* representation (see Figure 4.13) where dependent sources outputs λ_i conditioned on Y are represented in a clique. However, this version of Snorkel has not integrated the automatic computation of dependencies between the different sources λ_i, i.e. in the source code, all LFs are independent.

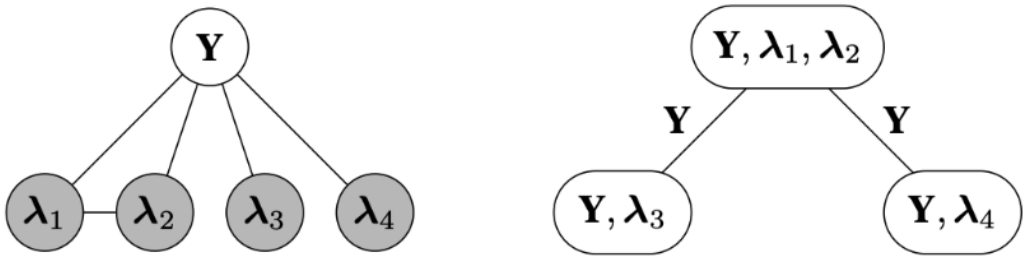


Figure 4.13 A weak supervision source dependency graph (left) and its junction tree representation (right).

This version of Snorkel is based on the work of (Loh and Wainwright, 2013) that uses the inverse covariance matrix of indicator variables on the vertices of a graph, to induce the conditional independence structure of the graph. It is therefore based on a class-conditional model of the LFs, where we assume a different LF accuracy for each different class – label value λ_i that each source emits (each source is modeled by more than a single accuracy

[5]If the multi-task learning (MTL) Snorkel MeTaL, whose project repository is in maintenance mode https://github.com/HazyResearch/metal, the simplified version, which is discussed in this section, is on the Snorkel team repository https://github.com/snorkel-team/snorkel

parameter). To learn the generative model over weak supervision sources, we define an indicator random variable for the event of a clique C taking on a set of values y_C:

$$\psi(C, y_C) = \mathbb{1}\{\cap_{i \in C} V_i = (y_C)_i\} \tag{4.28}$$

We therefore define $\psi(C) \in \{0,1\}^{\prod_{i \in C}(|Y_i|-1)}$, a vector of indicator random variables for all combinations of all but one of the labels emitted by each variable in clique C. The vector of sufficient statistics for the generative model we want to learn is $\mu = \mathbb{E}[\psi(C)]$.

To learn μ, Snorkel uses a matrix completion approach, where it considers two disjoint subsets of C; the set O of observable cliques — i.e. those containing labels from weak supervision sources — and the separator set S cliques of the junction tree — the unobserved latent variable $S = \{Y\}$. The covariance matrix of the indicator variables for $O \cup S$ is written in block form as follows (4.29 where T represent the multi-task setting):

$$Cov[\psi(O \cup S)] = \Sigma = \begin{bmatrix} \Sigma_O & \Sigma_{OS} \\ \Sigma_{OS}^T & \Sigma_S \end{bmatrix} \tag{4.29}$$

Following the work (Loh and Wainwright, 2013) which states that the inverse covariance matrix $Cov[\psi(C)]^{-1}$ is structured according to the dependency graph formed by the weak supervision sources, i.e. if there is no edge in the dependency graph between λ_i and λ_j, then the corresponding entries are O in the inverse covariance matrix. The inverse Σ^{-1} is of the form:

$$K = \Sigma^{-1} = \begin{bmatrix} K_O & K_{OS} \\ K_{OS}^T & K_S \end{bmatrix} \tag{4.30}$$

We can only observe $\Sigma_O = Cov[\psi(O)] \in \mathbb{R}^{nxn}$ (n represents the total number of weak supervision sources). The $\Sigma_{OS} = Cov[\psi(O), \psi(S)]$ is the unobserved block which is a function of μ, the generative model parameter we want to learn. Finally, $\Sigma_S = Cov[\psi(S)] = Cov[\psi(Y)]$ is a function of the class balance $P(Y)$, which is either known or has been estimated according to the unsupervised approach detailed in Appendix A.3.5 in (Ratner et al., 2019). Moreover, Σ_S is scalar, as it is the covariance of a single indicator variable for Y (see Appendix A.3.4 (Ratner et al., 2019)).

Given Σ_O, and Σ_S, the goal is to recover Σ_{OS} from which we can recover μ. We apply the block matrix inversion lemma to form the inverse covariance matrix of the observed cliques in the following manner:

$$K_O = \Sigma_O^{-1} + zz^T \tag{4.31}$$

where $z = \sqrt{c}\Sigma_O^{-1}\Sigma_{OS}$ and $c = (\Sigma_S - \Sigma_{OS}^T\Sigma_O^{-1}\Sigma_{OS})^{-1}$. Σ_O^{-1} is therefore the sum of a graph structured term (K_O) and a low-rank matrix (zz^T) (Loh and Wainwright, 2013) that represents marginalizing over the latent label Y.This leads to estimate z as a matrix completion problem in order to recover an estimate of μ (see Figure 4.14 where Ω represents the set of indices (i, j) where $(K_O)_{i,j} = 0 = (\Sigma_O^{-1})_{i,j} + (zz^T)_{i,j}$.

Algorithm 1 Source Accuracy Estimation for Multi-Task Weak Supervision

Input: Observed labeling rates $\hat{\mathbb{E}}[\psi(O)]$ and covariance $\hat{\Sigma}_O$; class balance $\hat{\mathbb{E}}[\psi(\mathbf{Y})]$ and variance Σ_S; correlation sparsity structure Ω

$\hat{z} \leftarrow \text{argmin}_z \left|\left|\hat{\Sigma}_O^{-1} + zz^T\right|\right|_\Omega$

$\hat{c} \leftarrow \Sigma_S^{-1}(1 + \hat{z}^T\hat{\Sigma}_O\hat{z}), \hat{\Sigma}_{OS} \leftarrow \hat{\Sigma}_O\hat{z}/\sqrt{\hat{c}}$

$\hat{\mu}' \leftarrow \hat{\Sigma}_{OS} + \hat{\mathbb{E}}[\psi(\mathbf{Y})]\hat{\mathbb{E}}[\psi(O)]$

return ExpandTied($\hat{\mu}'$)

Figure 4.14 Algorithm approach to the matrix completion problem to estimate the generative model parameter μ.

We have just seen how to estimate the parameter μ of the generative model, which is also called the *label model* in this second Snorkel version, with a class-conditional covariance matrix approach using the observed conditional dependency graph of the weak supervision sources. This is a very interesting approach (faster and clearer (better written) than the first version) since it depends on the structure of the graph formed by the observations (LFs outputs), and not on the number of data points.

However, this dependency graph is not automatically estimated in this version (neither in Snorkel MeTaL nor in the stable version source code); users can manually provide the conditional dependency structure, but the model won't take them into account. We first tried to add the dependencies generated by the first version into this model. We soon became aware that we needed to implement additional methods to integrate them. It was first necessary to add the triangulation step to create the junction tree from the conditional dependency graph (see Section 4.1.3 for details on the Triangulation step). Then we had to integrate these cliques (dependencies) in the matrices, however, to avoid some errors in the program, it was necessary to add a small value to the diagonal of the K matrix which did not make sense in the model. This conditional dependency structure could also be estimated using the same covariance matrix approach as explained in (Varma et al., 2019). However, we had the same errors when it came to including them in the matrices.

We seek to improve the performance of our LFs that abstain on some data points by using dependencies to increase recall. Unfortunately, the assumptions used in the junction tree model that works in some conditions are tricky. This is probably why dependencies are not implemented in the last version of Snorkel. Even if the calculation time is slower in factor

graph modeling (due to message propagation in the whole graph for every factor), we obtain an accurate calculation of several types of dependencies. With junction trees and covariance matrices, we can only distinguish whether the LFs are correlated or not, so as not to count them up twice (Varma et al., 2019).

When we published our results, the new version had not yet been released. But because of the importance of the dependencies for our work, we have continued our analysis using the original version of snorkel.

4.2.4 Discriminative Model

Once we've learned the LFs accuracies, correlations and dependencies, we get these noisy conflicting labels that we assign to the data and their associated confidence (the likelihood for a label of being true). The next step in the Snorkel pipeline is that we want to use these labels and train a Machine Learning model. We want to take advantage of all the rich structural data that we have access to, and then train another end Discriminative model using these labels. The second reason we want to train a second model is because LFs don't necessarily assign a label to every single data point. This discriminative model can retain the precision of the LFs while learning to generalize beyond the LFs, increasing coverage and robustness on unseen data. This is how Snorkel helps to get a large enough training data set to power modern deep models.

4.3 Conclusion

In this chapter we have first described a common formalism, graphical models, for representing independent relationships and computing conditional probabilities among a set of random observations. Graphical models make it possible to intuitively learn the structure of the data by studying the interactions of observations, and to perform efficient estimation algorithms faster, without worrying about the nature of each variable. We discussed the graphical models such as factor graphs or junction trees, which are handy tools in the context of probability distribution, particularly for non fully observed data.

In the second part of this chapter, we have introduced Snorkel which is a framework that allows users to significantly reduce the amount of work required in labeling data for machine learning applications. This project contains three main stages; the *Labeling Functions* (LFs) that encode all forms of weak supervision sources (knowledge bases, other trained models, domain expertise, crowd-sourcing etc.), the *Generative Model* that creates the noisy labels by unifying the conflicting and weak Labeling Functions outputs, and an *end Discriminative*

model that is our final classifier which is trained on the noisy generative labels. We explained the intuition behind the Generative model that uses a graphical model to estimate efficiently labels (annotations) for a large amount of unlabeled data. It models the correlations and dependencies of the weak supervision inputs in a probabilistic graphical model, and infers the marginal probabilities for each data point. If the recent version of the Snorkel system has a source code that is easier to understand, and is faster in time to infer the labels; nevertheless, the necessary modifications made for dependency estimation, made it very difficult to interpret the matrices' representation of the LFs, and thus the results of the generative model. Moreover, the representation of dependencies is more complete in the first version that uses factor graphs. We decided to keep the original version throughout our work.

The "tools" described in this Chapter are not only useful tools for our project, but their motivation and presentation allows us to understand the reason for their use. According to the project publications (Ratner et al. (2017b)), with the same amount of time writing the LFs and training an end model, Snorkel does better in terms of model performance than hand labeling data. The main reason is the scale that we can label data (five LFs can be applied to hundreds of thousands of data points), but we spend weeks or even months hand labeling data points. In our work, we would like to verify this performance with the difficult task of discourse structure prediction. We are going to address the set-up and design decisions we choose for our experiments in the following chapter.

Chapter 5

The Experiments

All classification attempts have to start from a theoretical position that tells one what there is to classify in the first place. For our approach to attachments this is specified, as already mentioned in Chapter 2, by theories of discourse structure—more specifically, the theory of discourse structure as formalized in SDRT (Asher and Lascarides, 2003). Our aim is to find attachments in multi-party conversations. We described the STAC corpus in Chapter 3 and motivated why we used it.

We also mentioned the problem of annotation campaigns: manual annotation is very costly in terms of time and human resources (linguistic experts). We then discussed the Snorkel framework in Chapter 4, which uses graph theory in its generative model to be able to annotate a great amount of data quickly and efficiently without having access to ground truth labels.

In this chapter, we will explain how we used all these elements to construct our experiment for the prediction of discourse attachments. We will first explain our choices of pre-treatments performed on the STAC corpus. Then, we will explain how we adapted the snorkel framework to write our labeling functions. Finally, we will describe how we proceeded to evaluate these experiments.

5.1 Setting up the Working Environment

5.1.1 Data Preparation

We designed our weak supervision experiments on the "situated" STAC corpus (Asher et al., 2016, 2020; Hunter et al., 2015). We also wanted to compare our results to earlier work on multi-party discourse structure prediction on the same corpus, which was done by Afantenos et al. (2015); Perret et al. (2016) and more recently by Shi and Huang (2019). However, these

previous studies designed their prediction experiments on the linguistic-only version of the STAC corpus. And so we had to test our framework on STAC linguistic corpus as well as the situated corpus.

The corpus from (Perret et al., 2016), used by Shi and Huang (2019) is an early version of a "linguistic only" of the STAC corpus. It contains no non-linguistic DUs, unlike the STAC multimodal corpus. There is also on the STAC website an updated "linguistic only" version of the STAC corpus, which is the one we have been working with. It has 1,091 dialogues, 11,961 linguistic only DUs and 12,271 semantic relations (Table 5.1). The data set from (Perret et al., 2016) is similar to our linguistic only STAC corpus but is still substantially different and degraded in quality. Asher et al. (2016) report significant error rates in annotation on the earlier versions of the STAC corpus and that the current linguistic-only corpus of STAC offers an improvement over the (Perret et al., 2016) corpus used by Shi and Huang (2019). It also contains quite a few errors; for example, about 60 stories in the (Perret et al., 2016) data set have no discourse structure in them at all and consist of only one DU. These dialogues were removed in the recent only linguistic version we used, as these stories were obviously not a correct representation of what was going on in the game at the relevant point. The following two tables, table 5.1 and 5.2, illustrate the number of attachments per relation type on the two corpora we downloaded from the STAC project website[1] for our experiments.

We downloaded both corpora and reconstructed, from the incoming and outgoing relation files, a single table that contains the semantic relations between the DU pairs $\{(s,t,r)|s \neq t\}$ that stands for a link of relation type r from the DU source s to target t. All discourse annotations associated with the DU pairs are included as shown in the following fields list.

- **doc_id:** The game id.
- **dialogue_num:** The dialogue number.
- **relation_type:** The discourse relation r between the two DUs (s,t).
- **source** and **target_text:** text of the source and target DU
- **source** and **target_dialogue_act:** dialogue_acts of each DU
- **source** and **target_surface_act:** surface_acts of each DU
- **distance:** distance between the source and the target DU
- **source** and **target_turn_id:** turn number
- **source** and **target_emitter:** speaker of each DU
- **source** and **target_addressee:** DUs addressees
- **source** and **target_segment_type:** linguistic or non-linguistic segment

[1]https://www.irit.fr/STAC/corpus.html

Types	Counts
Acknowledgment	1543
Alternation	141
Background	86
Clarification_question	456
Comment	2037
Conditional	155
Continuation	1194
Contrast	537
Correction	232
Elaboration	1044
Explanation	527
Narration	103
Parallel	212
Q_Elab	645
Question_answer_pair	2914
Result	445

Table 5.1 The 12,271 relations in the only linguistic annotations.

Types	Counts
Acknowledgment	1771
Alternation	128
Background	134
Clarification_question	529
Comment	2529
Conditional	154
Continuation	9391
Contrast	539
Correction	293
Elaboration	1631
Explanation	567
Narration	69
Parallel	193
Q_Elab	661
Question_answer_pair	3865
Result	13741
Sequence	6863

Table 5.2 The 43,058 relation_types in the situated STAC corpus.

We formulate the attachment problem on the STAC multi-party dialogues (linguistic only and the situated corpus) as follows: given a dialogue (*dialogue_num*) that has been segmented into a sequence of EDUs, the goal is to predict directed attachments between the EDUs. We then had to construct all possible pairs of EDUs as our candidates for the attachment prediction task. The following section shows how we proceeded to construct these candidates.

5.1.2 Candidate Extraction

Candidates are the units of data for which labels are predicted. For this study, the candidates are all DU pairs, which could possibly be connected by a discourse relation. As mentioned in Chapter 4, Snorkel has a whole framework for building potential candidates from a document (the context hierarchy which breaks input documents down into sentences, spans, and entities for example when working on text). However, the context hierarchy functionalities are not adapted to our more complicated discourse segmentation task in this specific chat corpus. It is not straightforward to group by conversations without dialogue_num annotations, and to segment the dialogues into EDUs following a theoretical framework and therefore even less with this Snorkel functionality. We use our own method to create candidates from the annotated EDUs culled from the texts of the dialogues, making sure to limit the pairs to those that occur in the same dialogues.

Roughly 56% of the dialogues in the situated corpus contain only non-linguistic DUs. The discourse structure of these dialogues is more regular and thus less challenging; so we ignore these dialogues for our prediction task.

Following (Muller et al., 2012; Perret et al., 2016) we "flatten" *Complex Discourse Units* (CDUs), which remain challenging to predict. As discussed in Chapter 2, we adopted the same strategy by connecting all relations incoming or outgoing from a CDU to the "head" of the CDU, or its first DU that has no incoming link (see Figure 5.1).

98% of the discourse relations in both corpora span 10 DUs or less (see Figure 5.2). To reduce class imbalance and evaluate our work in comparison with previous work, we restricted the relations we consider to a distance of ≤ 10 as previously done (see Figure 5.3) ensuring that the graph structure in each dialogue is fully connected.

We added two other constraints to our candidate extraction step. First, we made sure that there are no backward links between non-player segments (Server/UI segments). We then ruled out the possibility of backwards relations between two DUs which have different speakers: it is linguistically impossible for a speaker of, say, an assertion d_1 at time t_1 to answer a question d_2, asked by a different speaker at time $t_2 > t_1$, i.e. before d_2 was asked. That is, we include (d_1, d_2) in our candidates but rule out (d_2, d_1). We keep backwards

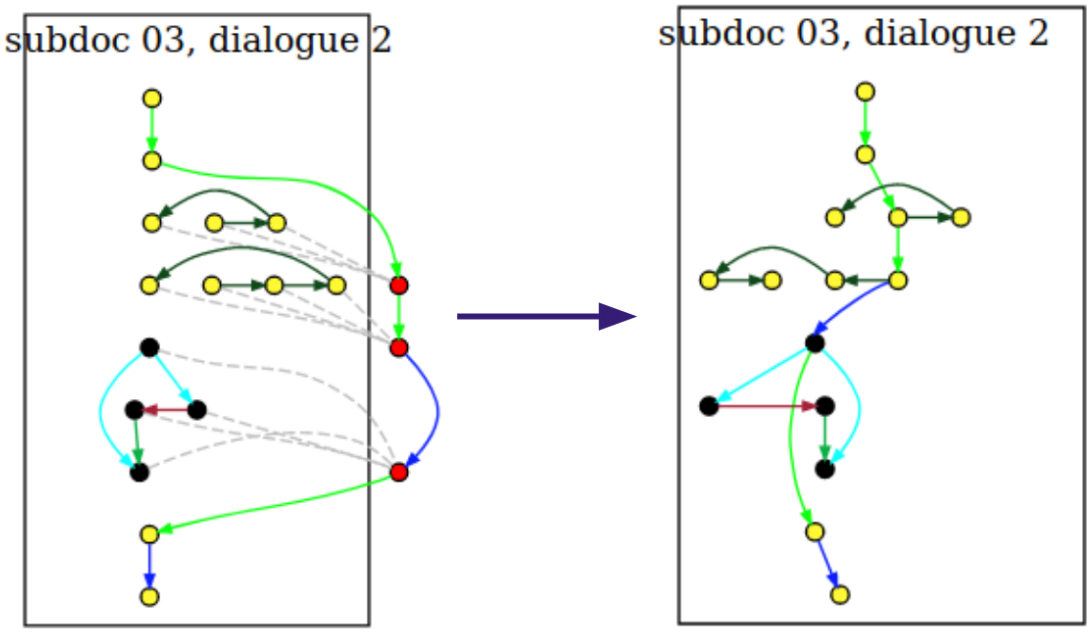


Figure 5.1 Flattening graphs process: we attached the incoming or outgoing links to the head of each CDU (red dot), the head being the segment of the CDU that has no incoming link.

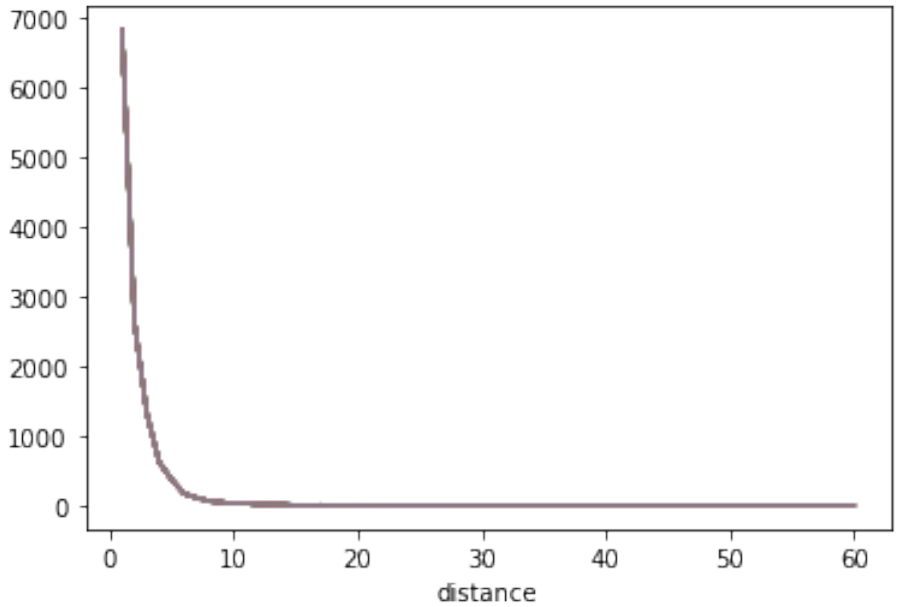


Figure 5.2 Distance plotting on the development set.

relations only within a single speaker turn — a single speaker turn can be extended, if a succession of turns are from the same speaker. This is when a player presses the Enter key after each piece of message.

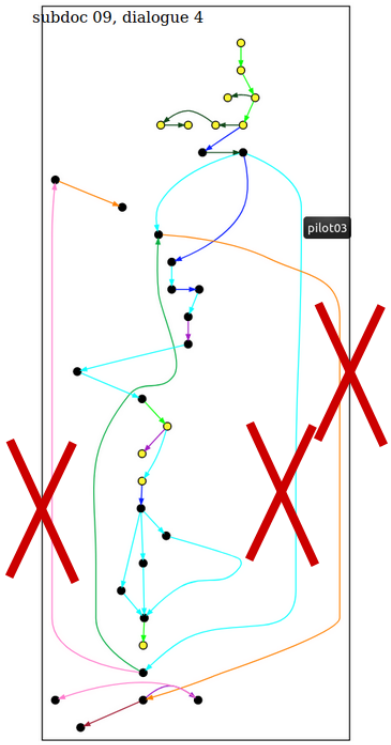


Figure 5.3 We do not consider relations that have a distance of 10 or more to reduce the sparsity problem and to compare the results to previous work.

We also performed the same operations on our version of the linguistic only corpus. These processes allowed us to generate the candidates on the two corpora as illustrated in Table 5.3:

	Candidates	Non-Attached EDUs	Attached EDUs
Target Situated STAC corpus	234600	204042	30558
Linguistic-only STAC corpus	76131	64037	12094

Table 5.3 Number of candidates of our weak supervision discourse attachment task.

The two tables, 5.4 and 5.5, show the remaining attachments by discourse relation type following our pre-processing. We have 12,500 fewer attachments in the situated corpus we used, the "Target situated STAC corpus". Whereas for the corpus containing only the linguistic segments, there is only a reduction of 177 attachments.

Types	Counts
Continuation	1162
Explanation	524
Background	83
Question_answer_pair	2901
Comment	2019
Acknowledgement	1506
Clarification_question	448
Contrast	524
Correction	227
Result	439
Elaboration	1035
Conditional	151
Narration	94
Parallel	206
Q_Elab	635
Alternation	140

Table 5.4 The 12,094 relations in the only linguistic annotations.

Types	Counts
Sequence	4086
Acknowledgement	1669
Clarification_question	495
Result	8644
Continuation	5456
Question_answer_pair	3628
Comment	2396
Correction	246
Contrast	503
Explanation	546
Narration	63
Conditional	149
Elaboration	1597
Q_Elab	644
Alternation	127
Parallel	178
Background	131

Table 5.5 The 30,558 relations in the Target situated STAC corpus.

The Candidate Split

The corpora we used are divided into a development (DEV), train and test set (Table 5.6). The development and test sets are each 10% of the total size of each corpus. We took the same data sets (the same set of game_id) as the previous work on linguistic only STAC corpus (Afantenos et al., 2015; Perret et al., 2016), but we chose to take the DEV set as part of the training set (Figure 5.4). Our approach is to automatically annotate the entire training set, based on our understanding of part of it (the DEV set).

	Target Situated STAC		**Linguistic-only STAC**	
	non-attached	attached	non-attached	attached
Train	186198	27803	58671	10980
Dev	27310	3868	10798	1794
Test	17844	2755	5366	1114

Table 5.6 Number of candidates per train/dev/test set.

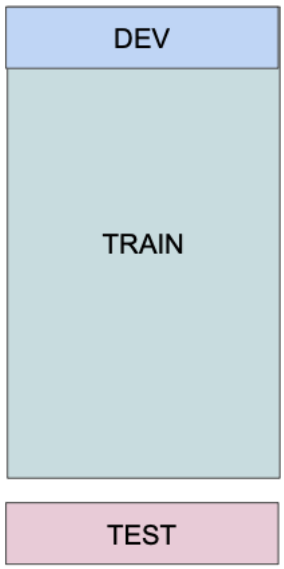


Figure 5.4 Sets distribution.

5.2 Developing and Adapting Snorkel Tools

In building the rules for the generative model for our task, we used the definition of the SDRT discourse relations in the STAC corpus, but we also looked at the annotations of the

DEV set, to identify particular patterns. We have chosen to look at the manual annotations of the DEV set, because for each task, it is important to look at the data we want to analyze, and to produce a number of annotations to capture the essential elements we want to learn – here the important patterns for discourse attachment. In this section, we will explain how we have designed these rules. Our rule set and their description are available in https://tizirinagh.github.io/acl2019/.

5.2.1 Labeling Functions

The choice of rule types

We wrote a set of labeling functions (LFs) to apply to the candidates X. Each LF λ makes an attachment decision for a given candidate x: it returns a 1, a 0 or a -1 ('attached" / "do not know" / "not attached").

In Chapter 2 and 3, we discussed complex structures, such as the non-treelike structures and in particular the so-called *double relations*. This representation is characterized by a DU that has two incoming links from the same source DU, but with different labels. Consider the following example (1) (Graph visualization link) composed of two EDUs. They have been annotated with two relations in STAC: Contrast(a,b) and Comment(a,b).

(1) amycharl: [it s not your fault,]$_a$ [but grr]$_b$

To help us in building attachments, we wanted to conceptualize the dependencies between discourse relations for attachment. The idea is to allow the generative model of Snorkel to find the dependencies between our LF types. Each of our LFs is written and evaluated with a specific relation type in mind. In this way, LFs leverage a kind of type-related information, which makes sense from an empirical perspective as well as an epistemological one. An attachment decision concerning two DUs is tightly linked to the type of relation relating the DUs: when an annotator decides that two DUs are attached, he or she does so with some knowledge of what type of relation attaches them. Thinking of attachment in the abstract was rather unintuitive; as human experts and as designers of the rules, we thought of the attachments with a discourse relation in mind. We have chosen the following discourse relations: *Result*, *Question-answer-pair (QAP)*, *Continuation*, *Sequence*, *Acknowledgement*, *Conditional*, *Contrast*, *Elaboration* and *Comment*. These are the most frequent relations therefore those that cover the most attachments, but also the most important ones for understanding multi-party dialogues.

Attachment categorization between linguistic and non-linguistic EDUs

NL_NL attachments

There are four different types of EDU pairs in our situated corpus: "NL_NL" pairs (non-linguistic source → non-linguistic target), "NL_L" or "L_NL" pairs (non-linguistic source → linguistic target, or linguistic source → non-linguistic target), and linguistic-only pairs ("L_L" for linguistic source → linguistic target). To construct our LFs, we first focused on the most regular links; the attachments between the non-linguistic segments, "NL_NL" (non-linguistic source → non-linguistic target). These are based on the game rules[2].

We first built lists of keywords, such as the resources that players use in the game:

```
resources = ["clay", "ore","sheep", "wheat", "wood", "knights"]
card = ["Soldier", "Year of Plenty", "Road Building", "Monopoly",
    "development"]
building = ["settlement", "road", "city"]
bank = ["bank", "port"]
```

We also constructed generic forms of the logs —game messages produced by the system — displayed in the game interface using simple regular expressions (regex) called by our LFs, which we have referred to as the "helper functions".

```
def XplayedAPCard(text):
    if re.search(" played a " +
                 ("|".join(card)) + " card", text) is None:
        return False
    else:
        return True

def traded(text):
    if re.search(" traded [0-9]+ " +
                 ("|".join(resources)) + " for [0-9]+ " +
                 ("|".join(resources)) + " from", text) is None:
        return False
```

[2]https://www.catan.com/service/game-rules

```
    else:
        return True
```

We then constructed our attachment rules, consulting the DEV set, sorting them by discourse relations, and using the *helper functions*. Some examples of these attachments that translate the non-linguistic moves in the game are the sequence of moves that end and begin a dialogue in our corpus, the resource distributions made after a player has rolled the die, the trade moves followed by an accept or reject of these trades, the end of turn for a player and the invitation of another player to roll the dice.

(2) X ended their turn - [result] - It's Y's turn to roll the dice

(3) X made an offer to trade M R1 for N R2 - [QAP] - X traded M R1 for N R2 from Y

Below is the source code 5.1 written in Python for Example (3) where we make sure that the name *X* of the player mentioned at the beginning of each of the two EDUs (source and target) is the same.

```
def LF_QAP_NL_NL_case1(row):
    l=0
    if madeanoffertotrade(row.source_text)
    and traded(row.target_text) \
    and row.source_emitter == "Server"
    and row.target_emitter == "Server":
        if row.source_text.split(' ')[0] == row.target_text.split(' ')[0]:
                l=1
    else:
            l=0
    return l
```

Code 5.1 Example of a NL_NL attachment rule.

As discussed in Chapter 2, we had to flatten the SDRT structures because we could not handle the CDUs in relation to the rest of the graph. This flattening process led to paying attention to the head of CDUs. We built a *helper function* to check if a segment is either the first or the only segment in a turn, to ensure that we attach to the head of the CDU. These operations have been used for "NL_NL" attachment rules easily, but for linguistic

only rules, it was more complicated. Figure 5.5 shows the changes made in Example (5) to accommodate the flattening process for Example (4).

(4) X rolled an M1 and M2 $_A$ - [result **CDU**] - [Y gets N1 R1s $_B$ - [continuation] Z gets N2 R2s $_C$]

(5) X rolled an M1 and M2 $_A$ - [result] - Y gets N1 R1s$_B$ - [continuation] Z gets N2 R2s$_C$

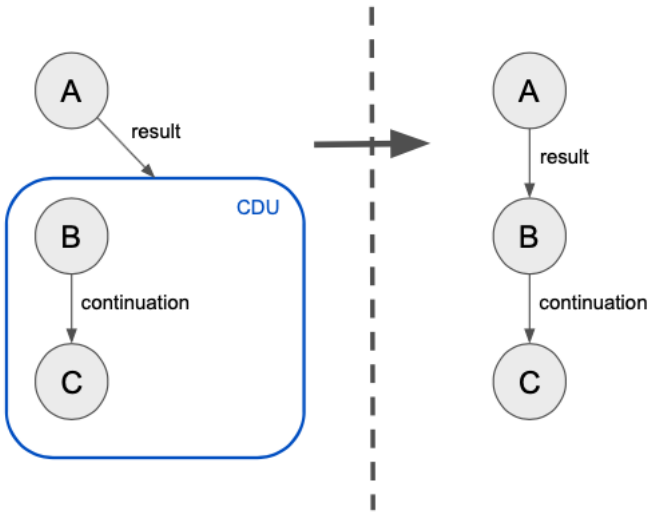


Figure 5.5 CDU elimination.

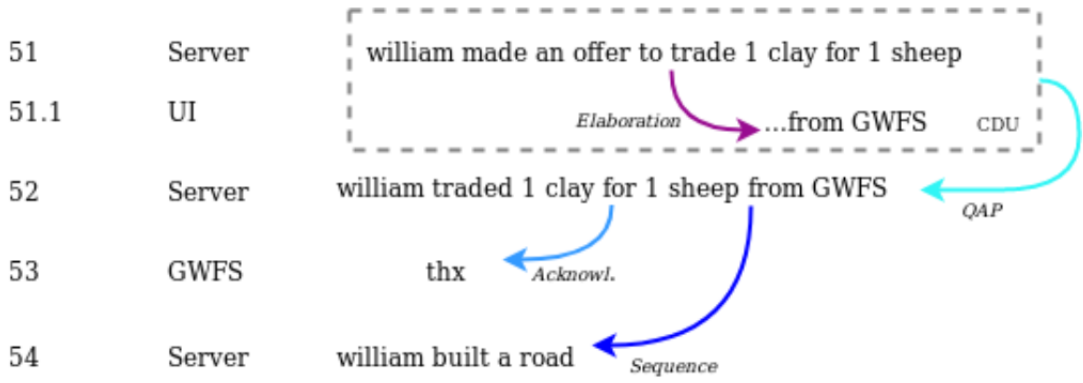


Figure 5.6 Relations between EEUs.

The exchanges in the situated STAC corpus are broken down into turns that begin when a player gets the dice and end when he finishes his turn and passes the dice to the next player. These events are related with Result, Continuation and Sequence relations. EEUs are also

frequently arguments to QAP and Elaboration relations (as seen in Example 5.6 (Click here for full graph)). The final outcome is that we wrote NL_NL rules in the *category* Result, Continuation, Elaboration, QAP and Sequence. These NL_NL LFs form the events of the game in which the EEUs are semantically connected.

NL_L / L_NL attachments

We examined attachments between linguistic and non-linguistic segments in the DEV set. We generally find links beyond the boundaries of the game turn, such as when a player comments on a game move from the previous turn (as seen in Example 5.6 when GWFS thanks in turn 53 the trade william made with him in turn 52). In the *non-linguistic* source and *linguistic* target attachments, "NL_L" direction, we are often dealing with comment or acknowledgment relations. While the opposite direction, i.e. "L_NL" attachments (linguistic source → non-linguistic target), is more often a cause-effect or a Sequence relation.

(6) inca: see you :) - [result] - Server: inca left the game

(7) jon: nice game though - [sequence] - UI: jon ended their turn.

```
def LF_Acknowledgement_NL_L(row):
    #non-ling --> ling
    if ( ( itsXsTurnToBuildP(row.source_text)
           or satDownAtSeat(row.source_text)
           or wonthegame(row.source_text)
           or traded(row.source_text)
           or joinedTheGame(row.source_text) )
         and (any( x in row.target_text.lower()
                   for x in acknowl+opinionWords+helloWords) )
         and (row.source_emitter != row.target_emitter)
         and (row.distance <= 9)):

             l=1
    else:
             l=0
    return l

def LF_Result_NL_L(row):
```

```
#non-ling --> ling
if ("type *addtime* to extend this game another"
    in row.source_text.lower()) \
and ("addtime" in row.target_text.lower()):
     l=1
else:
     l=0
return l
```

Code 5.2 Example of NL_L attachment rules.

As shown in "LF_Acknowledgement_NL_L" in the Code 5.2, our rules exploit the distance between pairs of segments, constraints that we observe in the development set. We tried to constrain the long distance attachment for some links (especially for acknowledgment or comment relation) to try to reduce false positive attachments.

L_L attachments

We constructed several rules for linguistic-only attachments according to the 8 discourse relations we chose: *Result*, *Question-answer-pair (QAP)*, *Continuation*, *Acknowledgement*, *Conditional*, *Contrast*, *Elaboration* and *Comment* (*Sequence* rules we wrote are only NL_L and NL_NL).

For the *Result_LFs*, we built a causal link by exploiting the presence of explicit connectors such as those listed below. Without explicit markers, we look for a causal link between the expression of a wish (for a specific resource, for example) and another player's response to grant those wishes.

```
resultWords = [ "so", "accordingly" , "as a result", "consequently",
                "hence", "in the end", "now that", "then", "thereby",
                "therefore", "thus"]
```

Offers and counter-offers are often in the form of questions in our chat corpus. We have therefore used Dialogue acts and Surface acts annotations[3] to link "Question, Request,

[3]Due to lack of time before submitting our results, we used these manual annotations which we later replaced with regular expressions that only take into account the raw text of the input segments (see Chapter 7).

OFFER or COUNTEROFFER" to "ACCEPT or REFUSAL" segments. We have also exploited the addresses annotations in this category of *Question-answer-pair (QAP)*' rules.

For the *Continuation* rules, we constrained this type of attachment in two ways; first, we added a constraint to attach two close segments (distance less than or equal to 3) which address the same person and whose target segment contains the *and* connector; second, we added another constraint to attach two close segments from the same player whose source segment contain comment words such as *mmm, yay, arrrrrgh, lulz, sorry, well, anyhoo, man, yup, ouch, lol, oh, ugh*, and the target segment is a question.

Acknowledgments are signaled by words like *OK, Right, Right then, Good, Fine*, etc.. It is difficult to determine what is being acknowledged, i.e. the first argument of the Acknowledgement relation, but it is also difficult to determine whether the acknowledgment signals an understanding or an acceptance of what was said. For these rules, we made sure that the emitters of the two segments were not the same, that if the addressee *X* is specified in the source segment, it is *X* that responds, otherwise that the target segment which contains the acceptance words is addressed to all players, "All", and the distance between the two segments is 1.

Conditional is often introduced by an *if... then*. To capture attachments that involve the discourse relation *Conditional* we have an attachment rule that attaches two nearby segments when they have the same speaker and the same addressee, and that the first segment start with "if" or "once".

But, however, unfortunately, on the other hand, nevertheless are all strong cues for *Contrast*. We made a rule that links an offer or counter-offer (Dialogue acts annotations) to a nearby target segment that contains the keywords for contrast.

Elaborations typically occur, when an agent makes an offer and then further specifies it. Sometimes the speaker will follow up one question with another. These questions often stand on an elaboration-like relation, as shown in example (9) from the STAC DEV set. In example (8) taken from the DEV set, *william* is specifying his offer given in (8-a).

(8) a. william: Does anyone have ore?
b. william: I have wheat

(9) a. ljaybrad123: Does anyone want sheep for ore?
b. ljaybrad123: Does anyone want 2 sheep for 1 ore?

We constrained the Elaboration attachment to successive segments that have the same emitters, that may have the same resources to exchange, the target segment which may be a question or a segment that contains expressions such as *I have, I will, give*, etc..

The STAC corpus is full of markers for *Comment*, like *sorry*, *Ooh*, *bah*, etc. Emoticons also indicate *Comments*. However, as with acknowledgment, it is very difficult to determine what is being commented on —i.e. the first term of the *Comment* relation. We only distance-constrained to 1 the pair of segments and token-constrained the target segment that contains comment keywords, emoticons or emojis.

We finally combined our rules by discourse relation type to form 9 Labeling Functions; e.g. for Acknowledgement we wrote 3 rules — two for L_L attachments and one for NL_L links — that we combined into a single LF, *LF_Acknowledgement*, that outputs 1 (*attached*) if one of the three rules outputs 1 for a given candidate, otherwise 0 (*abstain*). Only linguistic L_L LFs have been applied to the Linguistic-only STAC corpus, while all the rest of the LFs (L_L, NL_NL, NL_L and L_NL) were applied to the target situated candidates.

Intra-turn rule

As noted in (Perret et al., 2016), intra-turn segments are typically very locally attached. To capture this *Turn Constraint*, we have also written an *Intra-turn* LF that connects all segments within a turn, i.e. if the two EDUs are consecutive and from the same speaker turn, then the rule outputs 1 for a forwards link.

Hail-Mary rule

Our constraints on distances are not definite enough for long distance attachment and for non-attachment predictions. We therefore wrote the "Hail-Mary" rule,[4] which says that if all LF rules (the 9 LFs one for each relation type, plus the Intra-turn rule) return 0 (*abstain*) for a given candidate, then the Hail Mary rule outputs -1 (for $not - attached$).

5.2.2 The order in which the rules on candidates are applied

Candidates are pairs of dialogue units occurring in the same dialogue between which an attachment is possible, and are the units of data for which labels are predicted. The LFs use *local* information to determine whether the dialogue units in a candidate are attached, such as speaker identity, raw text, dialogue and speech acts and distance between units. Relying only on the local information of two considered EDUs/EEUs for the prediction of attachment in a whole conversation is a great disadvantage, because we know that attachments are

[4]Our Hail-Mary rule for non-attachment predictions refers to the expression used in American football, a "Hail Mary pass" which is a very long forward pass, usually made in desperation, with great difficulty of achieving a completed pass. Because of its low chance of success, it makes reference to the Catholic Hail Mary prayer for help.

conditioned by the surrounding discourse context. To address this drawback, we controlled the order in which *each* LF is applied to the candidates such that, the LFs are given access to some *global* contextual information, such as location in a dialogue, and what attachments have been predicted in the dialogue up to that point. We do not order the candidates between LFs— the LF for each relation type and intra-turn rule—, because otherwise we would interfere with the generative model that seeks to find the dependencies between these LFs.

Consider an example, where, given dialogue D with DUs A,B,C,D, appearing in that order, the set of possible candidates is: (A,B), (A,C), (A,D), (B,C), (B,D) and (C,D). We put the candidates in order of salient previous information, which for dialogue D yields the following ordered list L of candidates : $[(A,B),(B,C),(A,C),(C,D),(B,D),(A,D)]$. As a LF is applied to L, it first sees, for each target unit, the immediately preceding source unit, followed by the source units at incrementally increasing distances to the target unit. Chronologically speaking, an LF considers the immediate present before moving into the past of a dialogue, where it considers everything which took place before.

In other words, we ordered the DUs in a dialogue (all the DUs in a dialogue are ordered chronologically), then they are converted into an ordered list of candidates. We have a script for the order of application for each type of LF. For each pair of candidates in the generated ordered list, we only look at the candidates that are most likely to be attached given the relation type considered by the LF. Since we know that backwards relations only occur for Comment and Conditional relation types among the nine relation types that we implemented, for any other relation type we immediately register a -1 (for $notattached$) for each candidate where the target comes before the source. We then apply each of the LF rules, looking only at the candidates with DUs' linguistic or non-linguistic status we want to consider (we only look at linguistic $\rightarrow$ linguistic cases for "L_L" rules for example). As mentioned above, the outputs of all the rules of an LF are combined to give a single value ($1, 0$ or -1 for "attached", "don't know" and "not attached") per specific LF for a given candidate. Our label matrix is therefore an $(11 - by - m)$ matrix, where m is the number of candidates to label (see Figure 5.7).

The way that we apply the *QAP* labeling function differs from the other LFs. The QAP LF encodes two assumptions: first that for each question/source– answer/target pair, the answer segment is the answer to only one question segment in the dialogue; second, that each speaker can answer a question only once[5]. To illustrate the latter assumption, while a question like "does anyone have any wheat?" can be answered separately by all (non-asking) players, we will only count *one response* by each player. This is not always the case in

[5]We made this assumption because it is chat and people are not being so repetitive. They want to play the game.

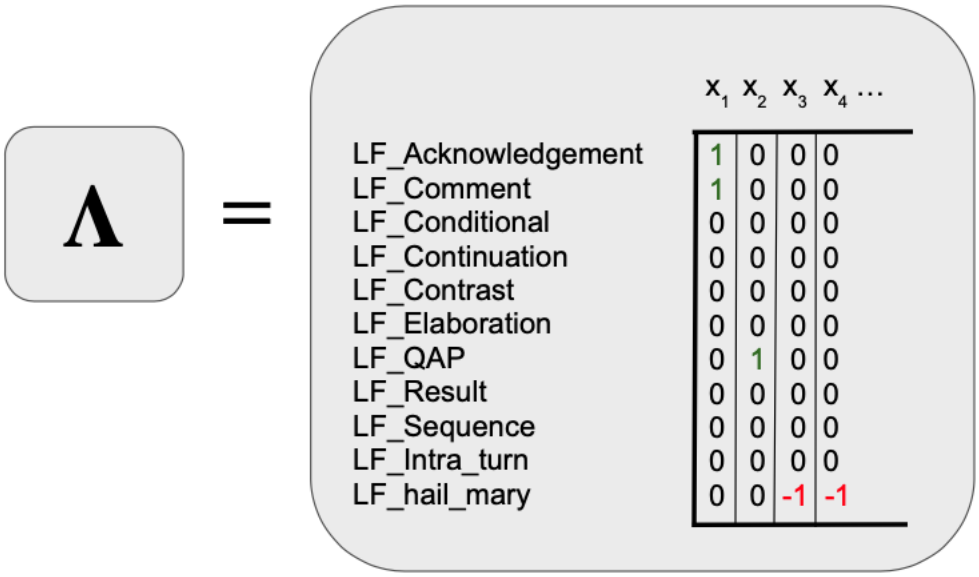


Figure 5.7 Our Label Matrix Λ.

the STAC data though multiple responses by one player were very rare. In effect, this was a simplifying assumption that allowed us to reduce false positives in the LF applications process.

We tested, improved and evaluated our LFs on the DEV set (our LFs performance and explanations are available here: https://tizirinagh.github.io/acl2019/).

5.3 The Mechanism for Predictions and Evaluations

5.3.1 The Generative Model

We are now ready to apply our LFs to as much data as we want, and to train the generative model with only the outputs of these weak supervision sources. We have not changed the parameters fixed by the authors for learning the structure (step size $= m^{-1}$, epoch count $= 10$, and truncation frequency $= 10$). However, we have changed some parameters for the generative model training; we have used a cross-validated grid-search over a parameter grid on the DEV set to select the number of training epochs (100) and the multiplicative decay of step size (0.95). We also fixed the gradient step size to $\frac{0.1}{m}$ (where m in the number of candidates labeled by the LFs).

When passing our $(11 - by - m)$ label matrix Λ to the generative model, it will return an m-D array representing the marginal probability of each candidate being True. The model

does not give binary outputs for binary settings, such as our attachment classification. We have not changed the model, but we have decided to calculate the threshold that gives us the best F1 score on the DEV set for attachment predictions, and apply it to the rest of the data (Train and Test sets) to have binary outputs (1 for attached, 0 for not attached). The best threshold is 0.85 ($p > .85$ for positive attachment) in the STAC DEV set.

Since we developed our labeling functions using the DEV set as a guide, and our generative model is composed of these labeling functions, we expect it to score very well on this set. We will use micro precision P, recall R, $F1score$ and *accuracy*, to evaluate our generative model which will be trained on the whole training set. We will apply several additional end extraction models (discriminative models) which will generalize beyond the DEV set, and which we will evaluate on a blind test set (i.e. one we never looked at during development).

5.3.2 The Discriminative Models

The standard Snorkel approach inputs the marginal probabilities from the generative step directly into a discriminative model, which is trained on those probabilities using a *noise-aware loss function* (Ratner et al., 2016). Ideally, this step generalizes the LFs by augmenting the feature representation - from, say, dozens of LFs to a high dimensional feature space - and allows the model to predict labels for more new data. Thus the precision potentially lost in the generalization is offset by a larger increase in recall.

We tested three discriminative models in our study. First, we tried a single layer BI-LSTM with 300 neurons, which takes as input 100 dimensional-embeddings for the text of each DU in the candidate pair (Figure 5.8). We concatenated the outputs of the BI-LSTM and fed them to a simple perceptron with one hidden layer and Rectified Linear Unit (ReLU) activation (Hahnloser et al., 2000; Jarrett et al., 2009; Nair and Hinton, 2010) and optimized with Adam (Kingma and Ba, 2014). Given that our data is extremely unbalanced in favor of the *unattached* class (*attached* candidates are roughly 13% of the candidates on the development set), we also implemented a class-balancing method inspired by (King and Zeng, 2001) which maps class indices to weight values used for weighting the loss function during training.

We also implemented BERT (Devlin et al., 2018)'s sequence classification model (source code on the link below[6]) with 10 training epochs and all default parameters otherwise (see Figure 5.9). BERT, the Bidirectional Encoder Representations from Transformers, is a text encoder pre-trained using language models where the system has to guess a missing word or word piece removed at random from the text. Originally designed for automatic translation

[6]Link to BERT sequence classification model code: https://github.com/huggingface/pytorch-pretrained-BERT/blob/master/examples/run_classifier.py

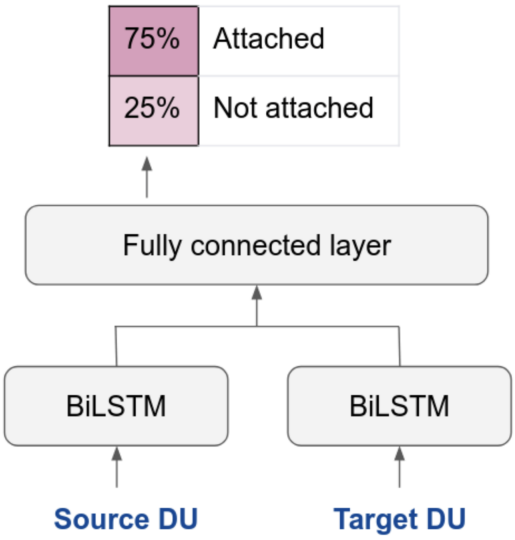


Figure 5.8 BiLSTM

tasks, BERT uses bi-directional self-attention to produce the encodings and performs at the state of the art on many textual classification tasks. In order to use BERT's sequence classification model, we had to binarize the marginal probabilities before moving to the discriminative step, using a threshold of $p > .85$ as explained in Section 5.3.1. Though this marks a departure from the standard Snorkel approach, we found that our discriminative model results were higher when the marginals were binarized and when the class re-balancing was used, albeit much lower than expected overall.

Finally, to facilitate comparison with earlier work, we also implemented a local model, LogReg* as mentioned in the following of the dissertation, that used marginal probabilities together with handcrafted features and a Logistic Regression classifier. Features are attributes that help the model learn (they can be specific words from the candidate's text itself or additional information inferred based on the original text). We applied the same features as in (Afantenos et al., 2015) (listed in their Table 2) in the discriminative LogReg model. This includes :

source/target Speaker_initiated_theDialog : If the source or target emitter initiated the dialogue,

source/target Speaker_first: First utterance of the emitter in the dialogue,

source/target_position: Utterance position in the dialogue,

distance : Distance between the pair of EDUs,

same speaker : If the two EDUs have the same emitter,

source/target_EndsExclamation : Ends with exclamation mark,

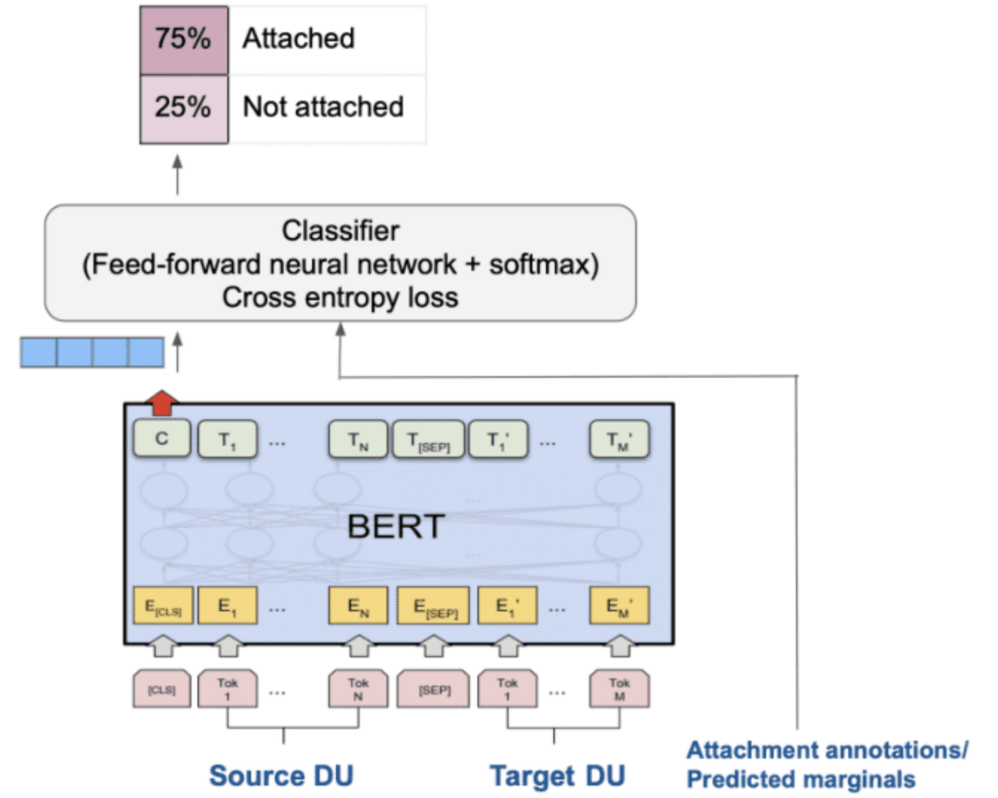


Figure 5.9 BERT sequence classification model

source/target_EndsInterrogation : Ends with interrogation mark,

source/target_possessive_pronouns: Contains possessive pronouns,

source/target_lexicons: Contains words from list of lexicons (“clay”, “ore”, “wood”, “sheep”, “wheat”, “brick”),

source/target_modal_modifiers: Contains modal modifiers (“could”, “should”, “may”, “might”, “can”, “must”, “will”, “would”),

source/target_question_words : Contains question words,

source/target_player_names : If the source or the target text contains a player’s name,

source/target_subject_lemma: Subject lemmas given by syntactic dependency parsing,

source/target_first and last : First and last words,

source/target_emoticons: Contains emoticons,

source/target_connectors: Contains connectors,

source/target_da and sa : Dialogue_act and surface_act of each segment.

We build the classifier by using the LogisticRegression module from sklearn. We fit the listed features into the classifier to train the model.

We also adopted these three models as a baseline (trained on the gold labeled data in the Training set of the STAC corpus), as well as a fourth model, *LAST*, which attaches every EDU in a dialogue to the EDU directly preceding it.

5.4 Conclusion

Predicting the discourse structure of a multi-party conversation is a difficult task. As we explained in Chapter 2, predicting attachment is crucial to building these connected discourse structures. But it is also a difficult problem for automatic processing. This chapter explains what we set out to do and how, to predict discourse attachment using the Snorkel *data programming* paradigm. We show how we modeled our prediction candidates, what kind of rules for the attachment labeling functions we chose, how we applied them on our structured candidates and we listed some of the final ((Badene et al., 2019b) version) written rules.

The Snorkel implementation of the data programming paradigm inspired our weak supervision approach for discourse attachment prediction. The LFs exploit information about the DUs' linguistic or non-linguistic status, the dialogue acts they express, their lexical content, grammatical category and speaker, and the distance between them—features also used in supervised learning methods (Afantenos et al., 2015; Perret et al., 2016). We wanted to create our own input label matrix for the generative model to mimic the flow of conversational events by controlling how our LFs apply on the candidates. This allows LFs to exploit information about previously predicted attachments and dialogue history in new predictions. We have presented the models we used for the discriminative step, as well as the baseline models trained on the gold manual annotations. The next chapter will present the results obtained.

Chapter 6

Snorkel Results

Most of the previous work on discourse structures prediction uses already annotated data to train their models (see Chapter 3.2). Such training sets are enormously expensive to create. In contrast with supervised approaches, we learn our models and the different weights on the classifiers without access to ground truth annotated data. Snorkel is designed so that no ground truth is needed, since we work in settings where access to good annotations (even a few) on which to train might be very expensive or difficult.

In this chapter, we will compare the results of our data programming approach, which has not been trained on the ground truth labels, with earlier supervision work on the same data. We will examine the set of probability-weighted labels that are generated from the combination of our handful weak supervision sources encoded in the LFs. We will investigate the Snorkel final models we chose that are trained on the generated labels.

6.1 Overall Results Analysis

We set out to test the performance of combinations of generative and discriminative models along the lines of the Snorkel data programming paradigm on the task of dialogue attachment structure prediction in order to automatically generate SDRT corpus data. While our results were consistent with what data programming promises—more data with accuracy comparable if not slightly below that of hand-labeled data—our most surprising and interesting result was the performance of the generative model on its own. As shown in Table 6.1 on *target situated STAC test data*, the generative model (GEN) dramatically outperformed our deep learning baselines—BiLSTM, BERT, and BERT + LogReg* architectures trained on gold labels—as well as the LAST baseline, which attaches every DU in a dialogue to the DU directly preceding it. In addition, stand-alone GEN also outperformed all the coupled Snorkel models, in which GEN is combined with an added discriminative step, by up to a 0.3 point

improvement in F1 score (GEN vs. GEN+BiLSTM). We did not expect this, given that adding a discriminative model in Snorkel is meant to generalize, and hence improve, what GEN learns.

	Precision	Recall	F1 score	Accuracy
SUPERVISED BASELINES				
LAST	0.54	0.55	0.55	0.84
BiLSTM on Gold labels	0.33	0.80	0.47	0.75
BERT on Gold labels	0.56	0.48	0.52	0.88
LogReg* on Gold labels	0.73	0.52	0.61	0.91
BERT+LogReg* on Gold labels	0.59	0.49	0.53	0.89
SNORKEL PIPELINE				
GEN + Disc (BiLSTM)	0.28	0.59	0.38	0.74
GEN + Disc (BERT)	0.49	0.40	0.44	0.86
GEN + Disc (LogReg*)	0.68	0.65	0.67	0.91
GENERATIVE STAND ALONE				
GEN	**0.69**	**0.66**	**0.68**	**0.92**

Table 6.1 Evaluations of weakly supervised (Snorkel and stand-alone GEN) and supervised approaches on the target situated STAC TEST data set.

Critical to this success was the inclusion of higher order dependencies in GEN and the fact that our LFs exploited contextual information about the DUs that was unavailable to the deep learning models or even the handcrafted feature model (see Table 6.2). GEN beats all competitors in terms of F1 score while taking a fraction of the annotated data to develop and train the model, showing the power and promise of the generative model.

One might wonder whether GEN and discriminative models are directly comparable. Generative machine learning algorithms learn the joint probability of X and Y, whereas discriminative algorithms learn the conditional probability of Y given X. Nevertheless, when we exploit the generative model we are trying to find the Y for which $P(X \wedge Y)$ is maximized. In effect, we are producing, *though not learning*, a conditional probability. So it makes sense to compare our generative model's output with that of other, discriminative machine learning approaches.

We also got surprising results concerning the supervised model benchmarks. Table 6.1 shows that LogReg* (Logistic Regression classifier described in Chapter 5.3.2) was the best supervised learning method on the target situated STAC data in terms of producing local models. This is evidence that handcrafted features capturing non local information about a DU's contexts do better than all purpose contextual encodings from neural nets at least on this task. We also implemented BERT+LogReg*, a learning algorithm that uses

BERT's encodings together with a Logistic Regression classifier trained on STAC's gold data with handcrafted features—in fact the same features as those from (Afantenos et al., 2015) and used in (Perret et al., 2016). BERT+LogReg* outputs a local model that improves upon BERT's local model, but it did not do as well as LogReg* on its own (let alone GEN), suggesting that BERT's encodings actually interfered with the correct predictions.

We will now look in detail at the results of each stage of the Snorkel pipeline, and we will also analyze the results on the linguistic-only version of STAC.

6.1.1 The Generative Step

Dependencies

GEN does not have access to the gold labels (attachments labels) on the Training set but uses the Training set as unlabeled data. It has only access to the 11xM Label Matrix Λ (M candidates *partially* labeled by our 11 labeling functions (LFs)). So in this model, the true class labels y_i are latent variables that are inferred from the LF outputs, which are estimated via Gibbs sampling over the Training set (80% of the STAC corpus + 10% of the DEV set), after it has been *partially* labeled by the LFs.

Generative Model on dev set	Precision	Recall	F1 score	Accuracy
Without higher order dependencies	0.45	0.70	0.55	0.87
With higher order dependencies	0.68	0.67	0.67	0.92

Table 6.2 Evaluations of positive attachment on Dev set with and without higher order dependencies.

If the default GEN model presupposes that the LFs are independent, this assumption does not always hold: one LF might be a variation of another or they might depend on a common source of information (Ratner et al., 2016). If we don't take these dependencies into account, we risk assigning incorrect accuracies to the LFs. Snorkel provides a more complex model that automatically calculates the dependencies between LFs and marginal probabilities which we use for the generative step. The higher order dependencies significantly improved the generative model's results on the STAC corpus (see Table 6.2).

Interpreting Marginal Distributions

When we obtain the results from the generative model GEN, we choose a threshold on the DEV corpus to apply to these marginals by calculating the threshold that gives us the best F1 score. The best threshold is 0.85 ($p > .85$ for positive attachment) in the target situated DEV

STAC corpus, where in the only-linguistic DEV set the threshold is 0.9. Figure 6.1 shows the probability distribution, on which even taking 0.8 as a threshold gives a lower F1 score because of false positive attachments. Binarizing these marginals allows us to evaluate GEN with respect to gold *attachment* labels on the STAC data.

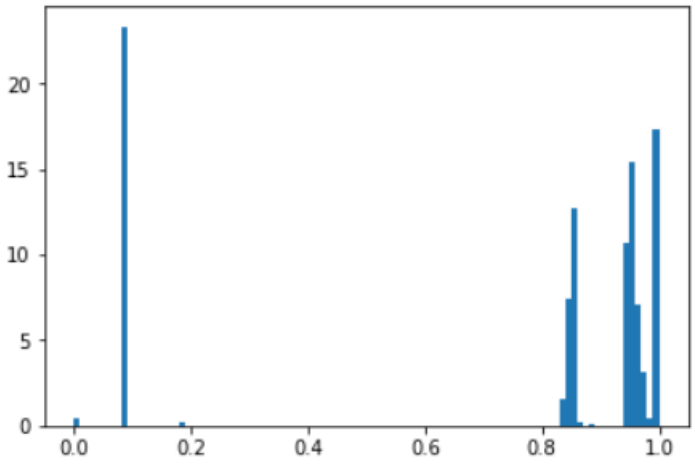


Figure 6.1 Probability distribution for positive attachment in the target situated STAC development set.

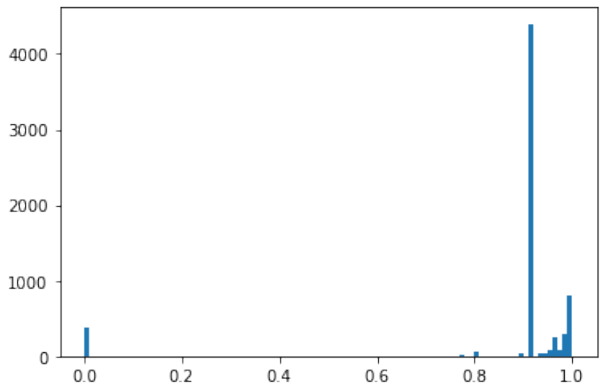


Figure 6.2 Probability distribution for positive attachment in the only-linguistic STAC development set.

Confused marginals' estimation would cluster at 0.5 in the histogram 6.1 and 6.2. Instead, we have a clear differentiation between 0.0 and 0.1. This shows that the generative model is very confident about the attachments to the right on the histogram, and not at all confident about attachments on the left of the diagram.

6.1.2 The Discriminative Step

We investigated GEN coupled with various discriminative models to test the standard Snorkel. Table 6.1 reports scores for various GEN-coupled discriminative models that take the binarized GEN predictions as input. In keeping with our analysis of the supervised benchmarks, we found that the best discriminative model to couple with GEN in the Snorkel architecture was LogReg*, far outperforming GEN with either a BiLSTM or BERT on the target situated STAC test set. Its results were only slightly less good than those of stand-alone GEN. We note that the intended result of the snorkel pipeline is a discriminative model, but our results suggest that it is better to stay with the generative mode. The dependency structure learned from our weak supervision sources seem to be good enough on its own to infer the latent variables.

In order to use the discriminative models we selected, we had to binarize the marginal probabilities before moving to the discriminative step, using a threshold of $p > .85$ on target situated STAC and $p > .9$ on linguistic-only STAC data as explained in Section 6.1.1. The standard Snorkel approach inputs the marginal probabilities from the generative step directly into a discriminative model, which is trained on those probabilities using a noise-aware loss function (Ratner et al., 2016). Ideally, this step generalizes the LFs by augmenting the feature representation - from, say, dozens of LFs to a high dimensional feature space - and allows the model to predict labels for more new data. Thus the precision potentially lost in the generalization is offset by a larger increase in recall. We observe this trend by comparing the results of LogReg* baseline model on the gold labels, and then GEN + Disc (LogReg*) in Table 6.1 (decreased in precision, but increased in recall). The binarization of GEN predictions marks a departure from the standard Snorkel approach, which may explain the unexpected results on Gen+Disc (BiLSTM) and on Gen+Disc (BERT) on the target situated STAC.

6.1.3 Comparison with Previous Work

The generative model shares with other *local* models (models presented in Section 3.2) the feature that it considers pairs of DUs in isolation of the whole structure. However, unlike other local models, our LFs enable the generative model to exploit prior decisions on pairs of DUs, and thus we exploit more global contextual information about discourse structure in the generative model than in our classical, supervised local models.

The two recent results of the work on the attachment prediction done by Perret et al. (2016) and Shi and Huang (2019), are obtained from supervised approaches that have been trained on a *previous* version of the linguistic STAC corpus, the Perret et al. (2016) version.

The deep sequential model in (Shi and Huang, 2019) decides the dependency links between EDUs incrementally, using local representations of the predicted path of EDUs. This model therefore uses jointly local and global information on attachment structure. If our LFs encode global information, the generative model classification is done on the DU pairs. So we can not really compare these two approaches, but it seems that this sequential model outperforms all the state-of-the-art baselines for dependency prediction (with $73,2\%$ F1 score) on the *previous* linguistic-only STAC data version. For comparison to the local model of Perret et al. (2016), which is closer to our classification approach on pairs of *EDUs*, we adopted the same LogReg* classifier as a baseline to predict attachment on the latest (corrected) linguistic-only STAC data. We have also decided to include BERT's sequence classification model as another baseline on the linguistic-only STAC data.

Local Models	Precision	Recall	F1
LogReg*	0.64	0.39	0.48
BERT	0.43	0.31	0.36
GEN	**0.58**	**0.50**	**0.54**
GEN + LogReg*	0.59	0.53	0.56

Table 6.3 Evaluation of positive attachments on *linguistic-only* STAC TEST data set.

The results reported by Perret et al. (2016) on local attachment with LogReg* classifier are 0.66 in precision, 0.38 in recall and 0.48 on F1 score. On our version of the linguistic-only corpus, we had a decrease in precision (of 0.02) and a slight increase in recall of one hundredth of a point with LogReg*.

In Table 6.3, we show how our model GEN and GEN + LogReg* on STAC test data compares to that of the linguistic-only STAC data. LogReg* outperforms the BERT model on the gold attachments labels (12 points difference on F1 score). Comparing GEN to LogReg* on the linguistic-only STAC data set, GEN has higher scores on F1 and recall than LogReg* model (through dependency estimation). However, we lose **6 points** in precision, which indicates that our LFs are not precise enough on the *linguistic-only* attachments. The results improved slightly when the discriminative LogReg* model was trained on the generated predictions. This shows that the parameters learned by the LogReg* model improved the coverage of the information that our LFs encode without loss of precision. This is what we aim by training a final model, the discriminative model, that can learn a more general representation.

6.2 Predicted Structures

The metrics we compared do not show the predicted structures. We therefore wrote a script to visualize the structure predicted by the binarized GEN model results. We found that these underlying graphs were not fully connected within a dialogue. Consider the small dialogue from STAC (click here for full structure visualization with relations) and its GEN predicted structure illustrated in Figure 6.3. The GEN graph is not fully connected. This may be because of the binarization process which does not take into account probabilities less than or equal to the threshold, thus potentially eliminating many attachments.

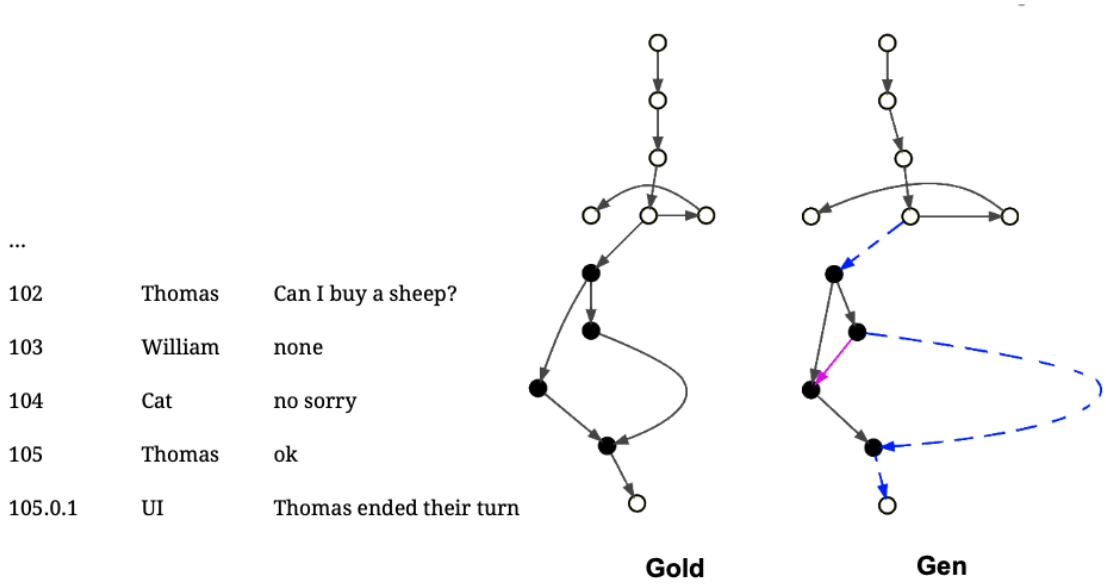


Figure 6.3 Comparison of the manually annotated Gold attachment structure and that predicted by Gen. The blue dashed arrows represent false negative (FN) attachment predictions and the magenta arrows represent FPs.

In addition to local information, our LFs include global structural constraints in the way they approach the candidates in the context of the dialogue before labeling them. The generative model performed better than (Perret et al., 2016)'s local model (in terms of F1 score). However, the structure generated by the Snorkel local results is not a directed acyclic graph (DAG), required by SDRT. We followed the previous work and implemented simple decoding steps with additional constraints in order to build a connected structure for each dialogue. We will compare our results with those of the global models from previous work.

6.2.1 Decoding Models

A set of highly accurate predictions for individual candidates does not necessarily lead to accurate discourse structures; for instance, without global structural constraints, GEN and local models may not yield the directed acyclic graphs (DAGs), required by SDRT.

MST and MST/short

As in previous work (Afantenos et al., 2015; Muller et al., 2012; Perret et al., 2016), we adopted the Maximum Spanning Tree (MST) algorithm, and a variation thereof, to ensure that the dialogue structures predicted conform to some more general structural principle. We implemented the Chu-Liu-Edmonds algorithm (Chu, 1965; Edmonds, 1967), an efficient method of finding the highest-scoring non-projective tree in a directed graph, as described in Jurafsky and Martin[1]. The algorithm greedily selects the relations with the highest probabilities from the dependency graphs produced by the local model, then removes any cycles. The result is a tree structure with one incoming relation per node. In cases of nodes with multiple equiprobable incoming relations, the algorithm takes whichever relation it sees first (the full algorithm is shown in Figure 6.4, where *CONTRACT* the cycle C means treat all nodes in the cycle as a single unit). When *contracting* the cycle C, the algorithm finds the incoming link *J* to the unit with the highest probability. Keeping the link J, which connects to the unit via node N, will remove any other incoming link to the node N. The algorithm then *uncollapses* or *expands* the cycle C to return the tree T and check again for cycles until no more cycles are found.

In addition, we implemented two MST variants *MST/short* and *MST/long* that always choose the shortest/longest relation (in terms of distance between DUs) among multiple high-probability relations.

MS-DAG and MS-DAG/short

Since SDRT structures can contain nodes with multiple incoming relations, i.e. are not always tree-like, we altered the MST algorithm in the manner of (Afantenos et al., 2015; Muller et al., 2012; Perret et al., 2016), forcing the MST to include all high-probability incoming relations which do not create cycles. This produces *MS-DAG* structures which are in principle more faithful to SDRT and dialogue structures. To implement this we ran MST algorithm 6.4, but kept track of all the links removed. Then we re-add each of these removed links one by one, checking for cycles each time. We do not re-add these removed links in any particular order.

[1] https://web.stanford.edu/ jurafsky/slp3/14.pdf

```
function MAXSPANNINGTREE(G=(V,E), root, score) returns spanning tree

    F ← []
    T' ← []
    score' ← []
    for each v ∈ V do
      bestInEdge ← argmax_{e=(u,v)∈ E} score[e]
      F ← F ∪ bestInEdge
      for each e=(u,v) ∈ E do
          score'[e] ← score[e] − score[bestInEdge]

      if T=(V,F) is a spanning tree then return it
      else
        C ← a cycle in F
        G' ← CONTRACT(G, C)
        T' ← MAXSPANNINGTREE(G', root, score')
        T ← EXPAND(T', C)
        return T

function CONTRACT(G, C) returns contracted graph

function EXPAND(T, C) returns expanded graph
```

Figure 6.4 The Chu-Liu Edmonds algorithm for finding maximum spanning tree in a weighted directed graph.

As well as for MST, we also implemented the *MS-DAG/short* and *MS-DAG/long* variants.

6.2.2 Decoding Step Results

Decoding on target situated STAC data

To further investigate comparisons between different architectures for solving the attachment problem, we compared various local models extended with the MS-DAG/short decoding algorithm, giving the global results shown in the right-hand columns of Tables 6.4 and 6.6.

With MS-DAG/short added, GEN continued to outperform the supervised approach LogReg* on *target situated STAC TEST data*. In Table 6.5, we experimented with adding all decoding algorithms to the local GEN results on *target situated STAC TEST* data set. This gave a boost in F1 score—2 points with classic MST and MS-DAG, and 4 points with the variants favoring relation instances with shorter attachments. The results of MST/long and MSDAG/long are similar to those of classic MST and MSDAG in terms of F1 score.

	Local Model			Global Model		
	Precision	Recall	F1	Precision	Recall	F1
LogReg* on STAC	0.73	0.52	0.61	0.65	0.64	0.65
GEN	0.69	**0.66**	**0.68**	**0.73**	**0.72**	**0.73**

Table 6.4 Comparison of local and global models (MS-DAG/short) on target situated STAC data.

MS-DAG/short improved the precision and recall of the GEN scores, although the boost was greater for recall than precision.

	MS-DAG	MST	MST/short	MS-DAG/short
F1 Score	0.709	0.709	0.726	0.726

Table 6.5 F1 scores of MST and MSDAG variants using GEN marginals as inputs.

It is not surprising that the decoding step improves the GEN results, since it eliminates some of the false positive relations that pass the generative threshold and includes some of the false negative relations that fall below the threshold. The general inverse power law distribution of discourse attachments (many short-distance attachments and fewer long-distance attachments) explains the good performance of the MST and MS-DAG shortest link variants on target situated STAC data. GEN + *MST/short* has the highest attachment score of all approaches to the problem of attachment in the literature (Morey et al., 2018), though we are cautious in comparing scores for systems applied to different corpora. This is what we will see in the following section, with the linguistic-only STAC data.

Decoding on linguistic-only STAC data

We also wanted to see how the decoding step fared on our version of the linguistic-only data set. With a decoding mechanism similar to that reported in Perret et al. (2016), LogReg*'s global model significantly improves over the GEN's. We see a 4 point loss in F1 score on GEN's global model relative to LogReg*'s, even though both used identical MS-DAG/short decoding mechanisms. This is what one would expect from a Snorkel based architecture, although it's not the rule that we observed for GEN. GEN did not beat LogReg* because it did not get a sufficient boost from decoding. We think that this happened because our LFs already contain some global information about the discourse structure, which meant that the decoding had less of an effect.

	Local Model			Global Model		
	Precision	Recall	F1	Precision	Recall	F1
LogReg*	0.64	0.39	0.48	**0.59**	**0.61**	**0.60**
BERT	0.43	0.31	0.36	0.46	0.47	0.46
GEN	**0.58**	**0.50**	**0.54**	0.58	0.54	0.56
GEN + LogReg*	0.59	0.53	0.56	0.59	0.55	0.57

Table 6.6 Decoding results (MS-DAG/short) of our methods on linguistic-only STAC corpus.

In comparison with the MS-DAG/short results on the target situated STAC corpus where GEN beats LogReg* (see Table 6.4), Gen+MS-DAG/short did not beat LogReg*+MS-DAG/short on linguistic-only data partly because the GEN's overconfidence (threshold to 0.9 on linguistic-only annotations) makes the rules too precise, which may explain why Gen don't gain much in recall with the MS-DAG/short decoding. Table 6.7 illustrates another important fact that the performance of the decoding algorithms depends on the type of data to which it is applied. On the linguistic-only annotations, the MS-DAG algorithms give the best results, while for the target situated annotations, both MST and MS-DAG provide the best score with their short variant (Table 6.5). The relatively higher performance of MST on the target situated annotations is arguably due to the fact that the situated structures are much more regular, while the structures in the linguistic-only annotations are more complicated and less predictable.

	MS-DAG	**MST**	**MST/short**	**MS-DAG/short**
F1 Score	0.548	0.44	0.452	0.558

Table 6.7 F1 scores of MST and MSDAG variants using GEN marginals as inputs on linguistic-only STAC data.

Comparison to previous decoding results

The two recent results of the work on the attachment prediction done by Perret et al. (2016) and Shi and Huang (2019) are reported in Table 6.8.

The MS-DAG/short decoding mechanism on the linguistic-only data set provided LogReg* only a boost of 12 F1 points, as seen in Table 6.6, which is significantly lower than what is reported in (Perret et al., 2016) with MST (see Table 6.8). This 12 point boost is the upper limit for boosts with MS-DAG/short that we were able to reproduce. This could be a result of our eliminating the degraded one EDU stories from the data set. What we

	Local Model			Global Model		
	Precision	Recall	F1	Precision	Recall	F1
Perret et al. 2016	0.66	0.38	0.48	**0.68**	**0.65**	**0.67**
Shi et al. 2019	-	-	-	-	-	**0.732**

Table 6.8 Decoding results on Perret et al. version of linguistic-only STAC corpus.

note is that taking into account the non-linguistic context is fundamental for the prediction of discourse structure. The results of Gen+MS-DAG/Short do not surpass Shi and Huang (2019)'s results, but are still very promising relative to our approach using weak supervision sources.

Connected Structures

While the best results are with MSDAG/short, the visualization of these structures is very difficult because many incoming links are taken into account. We could restrict the number of incoming links in the MS-DAG algorithm, but we want first to analyze those links that can form the diamond structures presented in Chapter 2 and 3 (analysis presented in Chapter 7).

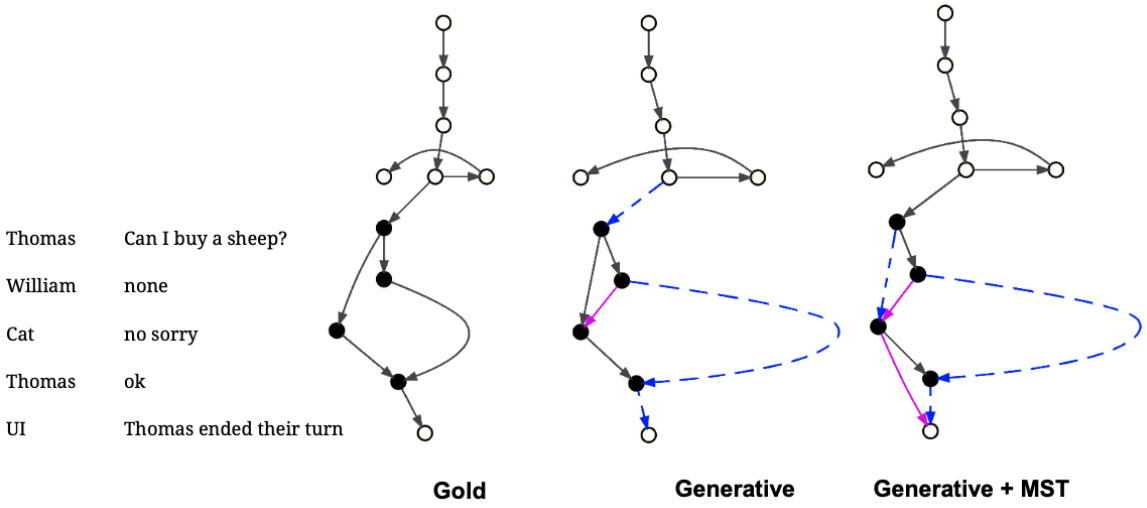


Figure 6.5 Comparison of the manually annotated, GEN binarized structure and that predicted by MST. The blue dashed arrows represent false negative (FN) attachment predictions and the magenta arrows represent FPs.

For better visibility, the Figure 6.5 shows the structure produced with MST algorithm. The decoding process has indeed produced a connected graph as we intended it to be. However, there are more FNs and FPs on the graph produced by the decoding step than the one

produced by GEN. This is due to the fact that the decoding model takes all the probabilities produced by Gen, whereas the Gen structure is based on the binarized scores. We note however, that the prediction errors, compared to the manual annotations, remain consistent (with the interpretation of the interactions) and allow long distance attachments. The false positive attachment between EDU [no sorry] from Cat and EDU [Thomas ended their turn] from UI, can be read as a result of cat's response [no sorry], without taking into account the diamond structure when Thomas acknowledges both [none] and [no sorry] responses. This is a violation of the right frontier, but it may be due to a higher weight of one LF compared to another. This is what we will seek to investigate in the next chapter.

6.3 Conclusion

In this chapter we have presented the potential of weak supervision for the prediction of discourse attachment. We show how our approach, which follows the Snorkel data programming paradigm, compares to other supervision models on the same data.

The most surprising result is that the generative model alone was able to outperform (or slightly underperform compared to GEN + the discriminative model LogReg* on linguistic-only data) all other approaches in terms of F1 score, suggesting that our 11 weak supervision sources described in Chapter 6 might be a good enough classifier on their own. The injection of higher-order dependencies into the generative model has been crucial to this success. We note, however, that GEN's results on the linguistic-only STAC data are not as good, which implies that our linguistic LFs (L_L rules) are not accurate enough.

The structures built by the GEN model are often not fully connected graphs, which is why a decoding step is desirable. It was important for the discourse structure to be complete within a single dialogue, as this is what our abstract model predicts must be the case for coherent dialogues. The decoding models we implemented, which are not included in the Snorkel pipeline, allows us to group the dialogue threads into a connected graph, structures that are consistent due to our rules which encode logical patterns. No wonder the results of the *short* decoding variants that choose the shortest attachment among multiple high-probability relations are better since discourse attachments in general follow an inverse power law (many short-distance attachments and fewer long-distance attachments). However, the constraints applied to the decoding are weak as the global model does not have access to the discourse features used by the local model.

In Chapter 7, we will further analyze the results and we will endeavor to improve some linguistic LFs to improve the discourse attachment precision scores.

Chapter 7

Results Exploration

The Snorkel approach facilitates error analysis because we have more control over the algorithms that are guided by expertise. We want to conduct an in-depth analysis of the results in order to understand the mechanism step by step and to be more confident about our model. Our goal is to be able to explain prediction errors and know how to solve them. In this way, we will bring transparency, explicability and feedback on our weak supervision approach.

In this chapter we will first look at which examples are predicted and which are not and why. We will also describe the refinements we have made to improve the precision of our linguistic LFs by focusing on two discourse relations; Question-Answering-Pairs and Acknowledgment. We will see how this affects the prediction results.

7.1 Quantitative and Qualitative Analysis

In Chapter 6 we have seen the performance of the generative model (GEN) which outperformed most of the machine learning models it was compared to. The decoding stage that we have implemented in addition to the Snorkel architecture has shown that the best decoding results were achieved by the short distant variant of MS-DAG.

In this section we will quantitatively explore the results of our predictions from the generative model (GEN) and the decoding MS-DAG/short mechanism (DEC) in all the training data that we have annotated automatically, and then we will interpret them qualitatively. We will be concentrating on two aspects; long distance attachments and complex predicted structures. We will proceed with a series of questions about our model's performance and then provide replies.

7.1.1 Error Analysis on Long Distance Attachments

What happened to the long distance links in the corpus?

We wanted to know if we were able to predict long distance attachments which are computationally difficult to determine, so we first counted the attachments with a distance greater than or equal to 2.

Data	TP	FP	Total
GOLD	9959	0	9959
GEN	4721	6605	11326
DEC	3186	1583	4769

Table 7.1 True and false positive long distance attachments.

The GOLD annotations have nearly 10,000 *long* distance links; nearly half of these are predicted in the Generative Model, while about a third of these are ultimately predicted in the MS-DAG/short decoding model (DEC). As explained in Chapter 6, the constraint of choosing the short distance attachment in our MS-DAG model is too strong, especially since we have no direct source information (linguistic or contextual) from the data. MS-DAG only chooses the shortest link among several same probability values. This explains the drop in DEC for true positive (TP) long attachments as shown in Table 7.1.

Among the FPs predicted by the Generative Model, there are also candidates not attached in the corpus, but which may be consistent given what transpired in the conversation. Many of these FPs are annotated attached by the LF which encodes the *Comment* discourse relation.

We have not written any rules for negative attachments (apart from "Hail-Mary"), but some of our rules contain distance constraints. There are 173,341 true negative (TN) long attachments in all GOLD candidate pairs. Among them, 166,736 were correctly predicted by GEN. This result is mainly due to our "Hail-Mary" rule, which predicted 165,504 TNs and 1,232 other cases are due to our constraints in the different LFs.

Can we find patterns in the contexts in which the true links were predicted (TPs) and those in which they were not (FNs) with GEN?

The Generative Model missed a lot of long distance links in Comments (1077 FNs), in Question_answer_pair (785 FNs) and Acknowledgement relations (725 FNs); but it did best with Result (1504), Sequence (1418) and Question_answer_pair (1125) (the first three relations where we find the most TP long distance attachments).

It is very difficult to know what someone is commenting on, especially when that person comments on an eventuality that did not happen just beforehand without explicitly referring to that eventuality. More than half of the unpredicted long-distance comments have a distance of more than 5. An example of an unpredicted long distance Comment is illustrated in (1) (Click here for full graph visualization). For this candidate, where there is a distance of 6 between the pair of segments, all LFs abstained, so only Hail-Mary rule predicted a non-attachment.

(1) 372 **Server:** Tomm rolled a 3 and a 5.
...
374 **Dave:** Thank god.

While it would be interesting to study the influence of interjections and extra information such as emojis in a conversation, it is even more important to identify the questions and their answers, as well as the acknowledgements.

GEN predicted more TPs (1125) than FNs (785) long-distance Question_answer_pair (QAP) attachments. The questions and answers should be the links that are most easily covered, we therefore have to find a way to reduce the 40% of FNs. We wrote rules mainly to align offers and counter offers using Dialogue acts and Surface acts annotations without restricting distances. The FNs in the QAP category are mainly unrestricted spontaneous questions/answers, as illustrated by example (2) (Click here for full graph visualization), where players coordinate to meet in Markus' office on a particular day and time.

(2) 704 **Dave:** any particular time you'd prefer?
...
706.1 **Markus:** I've no preference for Wednesday

The target segment was not annotated by the dialogue/surface acts that were taken into consideration in the LF. We need to write more global rules that will consider the general forms of questions and answers (see Section 7.2).

Acknowledgment attachments are very difficult to identify, yet are very important in multi-party conversations to track participants' agreement and coordination. GEN predicted a lot of long distance FNs (725) because our Acknowledgment rules had restricted the distance to 1 between the pair of EDUs. We assumed incorrectly in our rules that acknowledgements were close to what is being acknowledged. Whereas in the DEV data set there are more long distance acknowledgement attachments (123) than short distance ones (81). These are the Continuation, QAP and Result rules that attached 129 TP long distance Acknowledgment links in GEN.

How many TP long distance links in DEC were already in GEN and how many were new in DEC?

Nearly a third of long-distance attachments are predicted in DEC (see Table 7.1). 4705 *long* distance attachments were predicted in both GEN and DEC and of these **3179** are true positive ones.

All but **7** of the true positive long distance links in DEC were already predicted by GEN (long TP in DEC (3186) minus long TP that were preserved from GEN (3179)), so we should concentrate on the move from GEN to DEC.

These 7 cases are all 9 segments apart; 3 of them are Comment and one sample for each of Explanation, Elaboration, Sequence and Acknowledgment attachments. We have not implemented a rule for the Explanation attachment type. The Sequence case shown in (3) is an "NL_NL" link (non-linguistic source → non-linguistic target) which we have not considered in our rule:

(3) **UI:** Charlotte ended their turn.
...
Server: »> Less than 2 minutes remaining.

The other cases (Comment, Elaboration and Acknowledgment attachments) matched the written regexes, but as we constrained in distance the pair of segments, none of the GEN rules attached these candidates. It is clearly not a good idea to have distance constraints for attachments in multi-party dialogues, but it was the quick solution to limit false positives.

151 long distance candidates were predicted attached by DEC but not by GEN. Of these predicted links, only the 7 we saw above are true positives. DEC connected the graphical structure of the dialogues, thus taking into account the 7 TP long links (out of the 151 positive attachments) that GEN missed because of the threshold applied.

7.1.2 A closer look at complex structures

Why did we lose 1542 TP long distance attachments when we moved from GEN to DEC?

1542 TP long distance attachments were predicted by GEN but not by DEC. QAP and Sequence were the most frequently dropped and in 3rd position is the Acknowledgement relations. The Sequence attachments we missed are mainly non-linguistic ones that we did not take into account in our rules.

There are only m=1532 target segments in the long target set (TPs in GEN and FNs in DEC). We hypothesize that there were 10 cases in which the long distance links with the same target were dropped from GEN to DEC.

We want to consider all long distance links that were removed from diamonds or that were replaced by short distance links, even if the original long distance link in GEN was FP. There are 74 TP and 1258 FP short links (l,m) in GEN and DEC such that there is a long link (k,m) that is TP in GEN and FN in DEC. In the Example (4), which is FP short link (l,m), QAP and Comment LFs predicted a positive attachment (Click here for full graph visualization).

(4) 261 **Kersti:** which?
262 **Tyrant Lord:** oh wait

There is no short link (l,m) in DEC missing in GEN such that there is a long link (k,m) that is TP in GEN and FN in DEC. There are no long link (l,m) in DEC and missing in GEN either such that there is a long link (k,m) that is TP in GEN and FN in DEC. 198 FP and 2 TP long attachments had targets that were involved in long distance links (j,m), where source j is different from source k preserved from GEN to DEC.

Have we been able to predict correct diamond structures?

There are 238 diamond structures in GOLD annotations. None were fully predicted by DEC. There are only 12 out of 1868 fully correctly predicted diamonds in GEN (same head, tail and middle nodes in the graph). This shows the importance of the semantic content conveyed by the LFs in the generative model, even though the final scores have been thresholded. The global model constructs a DAG from the GEN probabilities but has no semantic information reflected.

An example of a diamond correctly predicted by GEN is the structure that contains the following segments (Click here for full graph visualization):

(5) 230 **Chameleon:** thoughts?
231 **skinnylinny:** Hmmm, do we want to? [230-Clarif-Question-231]
232 **Chameleon:** I am happy to keep playing [(230,231)-QAP-232]
233 **Nancy:** me too [232-Parallel-233] [230-QAP-233]
234 **skinnylinny:** fair enough [(232,233)-Acknowledgment-234]

A false positive (FP) attachment is predicted between EDU 231 and 234 (output from our Comment rule). The TP attachment between segments 233 and 234 is the consequence

of the two positive outputs of our Comment and Acknowledgment rules. The EDU peer attachments (230 - 231) and (230 - 232) are the result of the positive outputs of our Comment rule. These results reinforce our idea of modeling attachment through different types of discourse relations (see Section 5.2.1). However, we need to redefine the QAP type rules which do not cover many corresponding attachments and in addition are based on manual annotations.

We had also observed cases in which a question, Q, had (at least) two responses, R1 and R2, in GOLD annotations but R2 was attached to R1 in GEN rather than to Q. The example in Figure 7.1 (shown in Chapter 6) illustrates this phenomenon (click here for full structure visualization with relations).

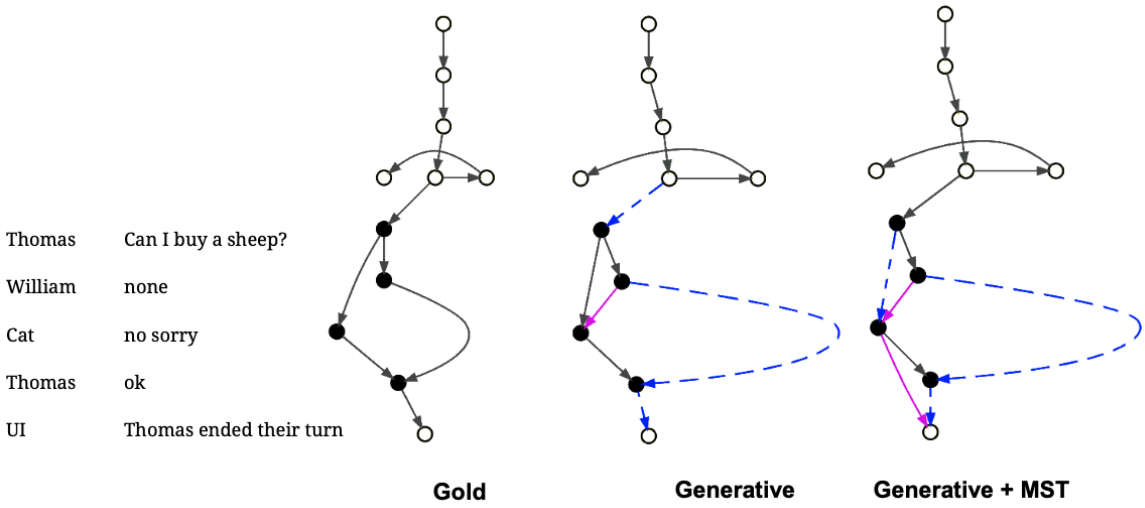


Figure 7.1 Comparison of the manually annotated, GEN binarized structure and that predicted by MST. The blue dashed arrows represent false negative (FN) attachment predictions and the magenta arrows represent FPs.

7.2 Two Case Studies

As mentioned in Chapter 5, we have used some manual annotations associated with individual EDUs/EEUs of the STAC corpus in our LFs. As for the distance, DU type (linguistic or non-linguistic segment) or the speaker names, this is information that we can easily have for the data we want to process, i.e. the meeting transcripts. We said in Chapter 5 that we can apply our LFs "to as much data as we want". This is not the case, since we need the annotations of addresses, dialogue acts and surface acts, which our rules exploit. My colleagues Thompson et al. (2019) have also successfully predicted the dialogue acts for

another conversational corpus using Snorkel framework. However, we spent some time editing the LFs on replacing the addressees and dialogue+surface acts annotations used and capture their patterns with regular expressions. This process did not change our scores in a significant way, but it did give us ideas for better expressing the discourse relations that the LFs convey.

The difference between the results on the target situated STAC data and the linguistic-only one presented in Chapter 6, indicated that our linguistic LFs (L_L rules) are more complex to define than the rules that encode the game moves. In Section 7.1 we have seen that many errors are related to Acknowledgment (ACK) and Question_answer_pair (QAP) links, discourse relations which are very important for the understanding of multi-party dialogues.

In this section we describe the changes we made for Question_answer_pair and Acknowledgment rules and we will see how this affects the scores on the two data sets.

7.2.1 Question_answer_pair Rules

When we wrote our rules (described in Chapter 5), we relied on the GOLD annotated *attachments* seen on the DEV set trying to cover as many attachments as possible. We did not evaluate our rules on attachments by *type of discourse relation* that the rules encode. Table 7.2 shows the generative model results of our QAP rules described in Chapter 5 on the linguistic-only data taking into account only QAP attachments, i.e. the labels are 1 for QAP attachments and 0 for no_QAP attachments.

Local Model	Precision	Recall	F1
GEN	0.61	0.68	0.65

Table 7.2 Previous results on linguistic-only QAP / no_QAP attachments on the DEV set.

We've written seven rules to cover QAP attachments relying on the annotations of the linguistic only DEV set and other intuitive principles:

1. **LF_QAP_1 :**This rule takes account of cases where the source is an EDUs of the "Offer" dialogue_act type and the target EDU is of the type "Refusal", "Counteroffer", "Accept" or "Request" dialogue_act. To do this we used game-specific regexes [e.g. "I can offer + *resources* + for + *resources*"] and elements from a resource list (sheep, rock, ore,...).

2. **LF_QAP_2 :** This rule takes account of the case where the source is a WH question and the target an answer, that does not have a question mark, at a distance less than or equal to 6.

3. **LF_QAP_3 :** This rule takes account of the case where the source_surface_act is of the form “Please choose...” and the target EDU is a response that take the form of opinions, comments, have emojis or that responds to a choice for the trade. It limits the attachments to a distance of 3.

4. **LF_QAP_4 :** This rule takes account of cases where the source is a WH question or an EDU that contains a question mark and the target is a response that takes the form of an opinion, a comment, has emojis or that responds to a choice for a trade. The distance is limited to 5 between source and target EDUs.

5. **LF_QAP_5 :** This rule takes account of cases where the source is an EDUs of the “Request” source_surface_act and target EDU is a response that takes the form of opinions, comments, has emojis or that respond to a choice for the trade. The distance is limited to 5.

6. **LF_QAP_6 :** This rule takes account of cases where the source EDU is a question formed using inversion (subject–auxiliary inversion) and the target segment is at a maximum distance of 5 (see Code 7.1).

7. **LF_QAP_7 :** This rule takes account of cases where the source EDU is a question that has one of the following words [“if anyone”, “anyone got”, “anybody have”, “anyone”], where the object to be exchanged appears in both source and target_text and where the answer is an opinion, decline or possession. It limits the attachments to a distance of 5.

8. **LF_QAP_8 :** This rule takes account of cases where the EDU source contains a question mark and where the EDU target does not have one. It limits the attachments to a distance of 5.

```
def subj_verb_inversion(phrase):
    subj_verb_inversion = []
    for token in phrase[:-1]:
        if (token.pos_ == 'VERB' and phrase[token.i+1].pos_ == 'PRON'):
            subj_verb_inversion +=  \
            [(str(phrase[token.i])+ ' ' + str(phrase[token.i+1]) )]
        elif ( token.pos_ == 'VERB' and
              phrase[token.i+1].pos_ == 'PUNCT'
              and token.i+2 < len(phrase)
```

```
                and (phrase[token.i+2].pos_ == 'PRON'
                     or phrase[token.i+2].pos_ == 'NOUN' ) ):
            subj_verb_inversion += [(str(phrase[token.i])+ ' '
                                     + str(phrase[token.i+1])
                                     + ' ' + str(phrase[token.i+2]) )]

    return(subj_verb_inversion)

def LF_QAP_LL_case6(row):
    l=0
    if any( subj_verb_inversion(nlp(row.source_text.lower())) ) \
    and (row.source_emitter != row.target_emitter)  \
    and ( row.distance <= 5 ):
        l=1
    return l
```

Code 7.1 An example of QAP rule with subject-verb inversion question.

As explained in 5.2.2 for all these LFs, we don't allow a speaker to answer his own question. We control the order in which each rule is applied to the candidates such that, given a particular segment x, the possible relations with segments closer to x with respect to textual distance are considered before relations with segments farther from x. This allows us to observe the history of the conversation on the discourse relation we want to apply. These new rules, which form the QAP_LF, have improved the scores as illustrated in the table 7.3:

Local Model	Precision	Recall	F1
GEN	0.74	0.70	0.72

Table 7.3 New results on linguistic-only QAP / no_QAP attachments on the DEV set.

7.2.2 Acknowledgment

GEN's results on ACK attachments are equal to zero as shown in Table 7.4. The ACK_LF did not predict much TPs (only 50 TP ACK attachments of 187 TP ACK attachments in DEV) compared to TNs, FNs and FPs (9863 no_ACK attachments) due to the distance constraint of 1.

Local Model	Precision	Recall	F1
GEN	0.00	0.00	0.00

Table 7.4 Previous results on linguistic-only ACK / no_ACK attachments on the DEV set.

We've written seven rules to cover ACK attachments relying on the annotations of the linguistic only DEV set and on the pairwise organization of some expressions which often go together (that we described in Chapter 3) such as when participants greet each other at the beginning of the game or when they say goodbye at the end of the game:

1. **LF_ACK_1 :** This rule accounts for cases where the source and target are greeting messages; it limits the attachments to a distance of 5 (see Code 7.2).

2. **LF_ACK_2 :** This rule models goodbye messages by limiting the distance to 4.

3. **LF_ACK_3 :** This rule takes account of cases where two pairs of EDUs from different emitters are at most 5 segments apart, that contain words from the ack_list such as the expressions ["ah", "excellent", "dommage", "fair enough", "fair point", "glad", "gotcha", "fixed", "damn", "perfect", "congrats", "good", "good game", "good luck", "great", "deal", "right", "done", "fine", "pleasure"].

4. **LF_ACK_4 :** This rule takes account of cases where two pairs of EDUs from different emitters, up to 7 segments apart, whose source segment does not contain a question mark and whose target segment contains words from the core_ack list such as the words ["agreed", "doesn't matter", "i can", "i did", "i see", "indeed", "k", "kk", "never mind", "no doubt", "no problem", "no worries", "not your fault", "sold", "sounds good", "sure", "me too", "my bad", "true", "fair enough", "will do", "ya", "yeah", "yeh", "yep", "yey", "you too", "sure thing"].

5. **LF_ACK_5 :** This rule accounts for an attachment between a source segment in the form of game-specific refusal or offer (e.g. "I can offer + resources + for + resources") that does not contain a question mark, and the target segment contains the expressions of the core_ack list. It limits the attachments to a distance of 7.

6. **LF_ACK_6 :** This rule is the same as rule 5, except that we check that the target segment contains emojis (we finally did not use this rule since it lowered the precision on ACK attachments even if the recall increases slightly).

7. **LF_ACK_7 :** This rule accounts for cases where the source segment does not contain a question mark or WH questions and the target segment of a different emitter is limited to 7 in the number of tokens it contains. It limits the attachments to a distance of 4.

8. **LF_ACK_8 :** This rule accounts for cases where the source segment does not contain a question mark or WH questions but contains at least one resource word and the target segment from a different emitter that contains the expressions of the ack_list or core_ack lists. It limits the attachments to a distance of 4.

```
greetings= ["hey", "bonjour", "hi", "bonjourno", "hola", "hello",
            "morning", "evening", "yo"]
byelist= ["bye", "cheers", "indeed", "thanks", "thank you", "yo",
          "see you"]

def LF_ACK_LL_case1(row):
    l=0
    if (any( t.strip() in
            re.findall(r"[\w']+|[.,!?;]", row.source_text.lower())
            for t in (greetings) ) ) \
    and (any( t.strip() in
             re.findall(r"[\w']+|[.,!?;]", row.target_text.lower())
             for t in (greetings) ) ) \
    and (row.source_emitter != row.target_emitter) \
    and (row.distance <=5) :
        l=1
    return l
```

Code 7.2 An example of new Acknowledgment rule.

Local Model	Precision	Recall	F1
GEN	0.21	0.27	0.24

Table 7.5 New results on linguistic-only ACK / no_ACK attachments on the DEV set.

These rules output -1 ("not attached") when the candidate is a backward link. The GEN evaluation of ACK_LF with the new rules has improved (Table 7.5). The improvement was mainly in TN and FP ACK attachments on the DEV set.

7.2.3 New Results

By only replacing the QAP_LF and ACK_LF described in Chapter 5 with the new versions described in this chapter, we significantly improve the precision of the generative model GEN on both corpora: by 9 points for the target situated corpus (Table 7.6) and by 5 on the linguistic only data (Table 7.7). These results highlight the power of weakly supervised methods to efficiently capture discourse local attachments. By applying the Logistic Regression LogReg* discriminative model on the GEN scores we have a slight improvement in recall particularly in the linguistic-only data.

	Local Model			**Global Model**		
	Precision	Recall	F1	Precision	Recall	F1
Previous GEN	**0.69**	**0.66**	**0.68**	**0.73**	**0.71**	**0.72**
GEN	**0.78**	0.58	**0.67**	0.74	**0.72**	**0.73**
GEN + LogReg*	0.76	**0.59**	0.66	0.72	0.70	0.71

Table 7.6 Comparison of previous (from Chapter 6) and new results of our methods on target situated STAC test data set (MS-DAG/short for the global model).

	Local Model			**Global Model**		
	Precision	Recall	F1	Precision	Recall	F1
Previous GEN	**0.58**	**0.50**	**0.54**	**0.58**	**0.54**	**0.56**
GEN	**0.63**	0.45	**0.52**	0.57	0.53	0.55
GEN + LogReg*	0.57	**0.48**	0.52	0.57	0.53	0.55

Table 7.7 Comparison of previous (from Chapter 6) and new results of our methods on linguistic-only STAC test data set (MS-DAG/short for the global model).

The MS-DAG/short decoding model lowered the precision of the generative model, but significantly improved the recall compared to the previous results from Chapter 6 (allowing us to reach the state-of-the-art score (Shi and Huang, 2019) on the target situated data, although this is not a direct comparison as Shi and Huang (2019) use the previous version of linguistic-only annotations (73,3% F1 score on target situated data)). However, compared to the results of the previous GEN raw, the *global model* scores improved by only 1% on the target situated data. On the linguistic only annotations, the MS-DAG/short decoding model scores decreased by 1%. Our simple and exogenous decoding model relies only on the probabilities of the generative model but does not encode global structuring discourse constraints to form SDRSs. We think we can put much more sophisticated decoding constraints (that we outline

in Chapter 8.2.1) that don't just work off local probabilities of arcs but on the full information about the arc in its discourse context.

We improved the true positives from 2473 to 2547 out of 3269 TP QAP attachments. While for ACK attachments, the TPs have not changed, but rather there is an increase in TNs and a decrease in FPs with our new ACK rules.

Let us see whether the added rules have improved the predictions we described in the first section of this chapter.

(6) 230 **Chameleon:** thoughts?
231 **skinnylinny:** Hmmm, do we want to? [230-Clarif-Question-231]
232 **Chameleon:** I am happy to keep playing [(230,231)-QAP-232]
233 **Nancy:** me too [232-Parallel-233] [230-QAP-233]
234 **skinnylinny:** fair enough [(232,233)-Acknowledgment-234]

The positive attachment output of our Comment rule is now reinforced by the new QAP LF for the attachment of segments 230 and 232.

We also observed the local attachment between answers, e.g. R1 to R2, rather than the attachment of long answers to the question, Q to R2. Our hypothesis is that local attachment won out over the other factors. Our rules have corrected these errors in the example shown in Figure 7.1 and also for other cases (click to see another simple example that starts with the question at turn 29).

More complex cases have also been solved where there are interwoven threads, or comments in the way of questions and their answers (click to see a more complex example that starts with the question at turn 86 of the dialogue). Adding QAP labeling rules helps override this preference of local attachment and improves the QAP predictions.

7.3 Conclusion

In this chapter we have given an example of how weak supervision can lead to a successful error analysis and how it naturally leads us to rules not only for attachment but also rules that enable us to assign discourse relation labels to attachments. We've also shown that refining the rules has led to modest improvements in our system's performance.

We concentrated on two relations. Question_answer_pair (QAP) and Acknowledgment (ACK) relations are essential in a semantic model of conversation to keep track of the participant points of view by linking their various contributions and acknowledgments. Through the different predicted examples, we found that the constraints modeled in the previous LFs were either too restrictive (distance to 1 for acknowledgments) or too broad

(based on the dialogues acts annotations for QAP). We improved these two LFs and found that the individual improvement for these types of relations was not proportional to the improvement in all attachments. Relying only on the local information of two considered EDUs/EEUs for the prediction of attachment in a whole conversation is a great drawback, because we know that attachments are conditioned by the surrounding discourse context. The constraints on the order of application of our rules are not sufficient for a complete discourse structure.

The MS-DAG/short decoding predictions described in this chapter show that the preference for local attachment won out over other factors and that there are many FPs because of the choice to include all high-probability links. We have observed what this simple decoding model could predict as discourse structure from the scores of the generative model. The FPs can be reduced by limiting the number of links to be included in the graph, but for a better representation of the discourse structure following SDRT, more sophisticated constraints have to be put in place, which we will discuss in the next chapter.

Chapter 8

Conclusion

The work presented in this thesis reflects the fact that the understanding of linguistic interactions is often based on common sense reasoning, for which knowledge of semantic relations and the structures to which they give rise is indispensable. In spite of the progress in machine learning in the field of Natural Language Processing (NLP) for practical applications of AI (Chatbots Language translator, Social Media Monitoring, Language Model in Automatic Speech recognition, Autocorrect, Autocomplete, targeted advertising), the need for a large amount of *annotated* data remains necessary but hard in practice to get. Our studies have shown that weakly supervised methods (Pan and Yang, 2009; Wang et al., 2020) allow both the automatic creation of sufficient training data containing complex language semantic information for machine learning, but also the creation of generative models that are competitive with the best that deep learning can offer. The data-driven programming method we explore allows us to model all kinds of logical linguistic constraints and to automatically learn dependencies between the available sources of information for the data we want to analyze.

In what follows, we review the main contributions of this dissertation by chapter before concluding with some remarks on future work and planned improvements.

8.1 Contributions

In **Chapter 1**, we described the context of our project from two angles: the linguistic and industrial contexts. We began by showing that when a speaker engages in conversation, it is important that her/his contributions can be easily followed by the listener or reader. To make a discourse coherent, it is natural to exploit linguistic tools—e.g., verb tenses, anaphora or the order of speaker turns (utterances)—to suggest how statements semantically connect to each other, as well as explicit transitional elements, such as conjunctions or other discourse

markers, that indicate these connections. We saw, however, that these explicit features are not sufficient individually; what matters is the reasoning about the content of the discourse units and the way in which the features interact. These discourse relations together with the discourse units they relate form a discourse structure that is essential to comprehension.

The next part of Chapter 1 motivated our interest in multi-speaker conversations in our industrial context. We explained that being able to infer discourse relations is crucial for the goals that we want to achieve in terms of conversation understanding. In particular, discourse structures for dialogues are beneficial for a variety of NLP tasks, including dialogue understanding, automatic summarization, sentiment analysis, and recommendations. Direct business applications are the Linagora Conversational Manager tool and LinTO meeting assistant.

The analysis of interactions between several agents is more difficult than the analysis of monologues because each agent has its own understanding and context. The emergence of communication platforms, and audio and video recordings of conversations offers a wealth of raw dialogue data to explore and process. This data gives us hope that we might find a way to combine discursive clues and other information available to us in order to predict discourse relations in a conversation, if only we can find a way to produce reliable annotations of the conversation data. At Linagora, our automatic speech recognition (ASR) team focuses on developing models for multi-speaker conversational speech recognition for meetings in order to provide transcripts that can be exploited to perform various tasks such as real-time recommendation or automatic summarization. As explained in Chapter 1, however, errors in automatic transcription, combined with the lack of discourse annotated data, the noisy nature of spontaneous conversational data and the complexity of multiparty dialogues present technical challenges that need to be addressed; the remainder of this dissertation has attempted to provide a solution to the problem of lack of annotated dialogue data.

In **Chapter 2**, we presented theoretical approaches to modeling discourse structure, focusing on the two leading theories, Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) and Segmented Discourse Representation Theory (SDRT) (Asher, 1993; Asher and Lascarides, 2003). We have explained how they identify relations between discourse units and construct the discourse structure that these units form. Building (dependency) discourse structures requires three steps in these formalisms: first, segmenting the text into the basic units of the discourse (Discourse Units (DUs)); second, predicting the attachments between DUs; and third, predicting the semantic type of the attachment. Theories of discourse structure differ with respect to the definition of discourse relations, the arguments (semantic or not) of these relations and their methods for constructing hierarchical structures. While much work has been based on RST, this theory is designed for written monologue; the theory does

not countenance non tree-like discourse structures and only allows discourse relations to arise between adjacent discourse units without reinterpretation of the structures. This does not allow the representation of dialogue structures which have a more complex form of their constituents. SDRT allows for a more general and readable graphical representation of the discourse structure and more precisely formalizes discourse relations. This is therefore the theory we decided to follow in order to build our dialogue parsing model.

In **Chapter 3** we presented, through different examples, the linguistic features of multi-party communication we wanted to analyze in the work presented in this dissertation. At this point, we described the corpus that we used for our project, namely the STAC corpus of multi-party on-line chats from an online version of the board game *the Settlers of Catan*. The STAC corpus is the largest corpus of annotated dialogues with complete discourse structures following SDRT (it is actually the largest corpus of dialogues annotated for discourse structure following *any* discourse theory). Moreover, it is the annotated corpus that most closely matches the meeting interactions we want to annotate (as explained in the industrial context in Chapter 1); there is a known goal during the exchanges between the participants (Asher et al., 2016; Hunter et al., 2018) and if the dialogues are asynchronous, the chat data can meander and go off topic; it is not edited to yield a single, clear thread, as most text documents are. However, while these annotations are helpful for developing models for dialogue parsing, they highlight a major problem with the annotation process: it took at least four years to produce them (Asher et al., 2016). Discourse structure annotation is a difficult task that requires linguistic expertise, which is costly and time-consuming to accomplish.

The effort required for manual annotation led us to explore new ways of annotating large data quickly and efficiently, which we explained in **Chapter 4**. We began our discussion by presenting graphical statistical approaches (Nielsen and Jensen, 2009) that allow us to estimate missing observations in data by estimating the dependencies of other known observations. We then introduced the data programming paradigm, along with an open source framework, called Snorkel, which consists of three main parts. First, there are Labeling Functions (or LFs) which are expert-composed functions that represent weak supervision sources on data points. Their distinctive feature is that these LFs can abstain when they do not observe the pattern they encode. Then comes the training data generation performed by the Generative Model which uses graphical statistical sampling approaches to combine all the valuable information given by the LFs, while at the same time mitigating the noise they may bring, to learn a noisy training set. The last piece of the pipeline is a Discriminative Model that is trained on the data annotated automatically by the Generative Model. It allows,

through other means than LFs, to learn other features on the data points to increase the coverage of the annotation results.

The more recent version of the Snorkel pipeline uses a matrix-based approach to represent the different outputs of the LFs that we needed to modify in order to implement the dependency estimation. Unfortunately, the necessary modifications made it very difficult to understand the label weights. Furthermore, there is no thorough exploration of the different types of statistical dependencies, whereas in the original version the dependencies are categorized (Similar, Fixing, Reinforcing, and Exclusive correlations (Ratner et al., 2017a)). Because the dependency estimation between the different sources is the most important aspect of this approach for our purposes, we decided to keep the original Generative Model of Snorkel throughout our project.

In **Chapter 5** we defined our task, what we set out to do and how, namely to predict attachment using the "data programming" paradigm with Snorkel. Predicting attachment is crucial to building connected discourse structures but it is also a difficult problem for automatic processing as attachments can be long distance. That is, a Discourse Unit (DU) will not necessarily be attached to the DU immediately preceding it, and can in fact be related to a DU introduced much earlier in the discourse. Another challenge stems from the sparsity of discourse connections: while there may be a great many candidate source-target DUs pairs in a dialogue, only a very small percentage of these candidates will actually be connected. A third challenge is that the question of attachment is a context sensitive one; whether a constituent is attached to another depends on other attachments in the context.

To allow for attachments in the whole dialogue, we generated all possible pairs (source-target) of DUs for each dialogue in the STAC corpus; these were the candidates on which we predicted attachment. To guide and facilitate the attachment task, we implemented some simplifying measures on the corpus, similar to previous work on the same corpus, which allowed us to evaluate ourselves by comparison with them. After motivating and describing these choices, Chapter 5 then presented our methodology in writing LFs for attachment based on the definition of discourse relations in SDRT, as well as the features available to us in the development set.

As explained later in the chapter, the difficulty of writing LFs to predict *non-attachment* led us to develop a "Hail Mary" rule which generates non-attachment outputs when other LFs abstain. In order to include global constraints in the attachment decisions and address the context sensitivity of attachments, we modified the Snorkel process to apply LFs to the data. We employed the context of what precedes in the discourse to apply our LFs to preserve tractability and exploit structure at the level of the sentence (description of DUs in our LFs) and the discourse (in the type of LFs and the application order of LFs). We chose to apply

each LF one at a time over every pair in the corpus, following an ordering determined by the chronological sequence of the DUs in a given dialogue. This allowed Snorkel find the dependencies between the different types of discourse relations, and thus the different types of LFs. We concluded Chapter 5 by presenting the discriminative models that we have tested, which we also used as baselines, as well as our evaluation process.

In **Chapter 6** we reported and analyzed the results of the attachment task with the Snorkel pipeline. While our results were consistent with what data programming promises—more data with accuracy comparable, even if slightly below, that of hand-labeled data—our most surprising and interesting result was the performance of the generative model on its own. For the baselines, we had the best F1 score with the model that used handcrafted features and logistic regression. When applying the Snorkel pipeline we also had the best score with the Logistic regression discriminative Model. Our results are significantly better than those of the baselines or those presented in previous work (Afantenos et al., 2015; Perret et al., 2016). The Discriminative Model did not improve the results of the Generative Model, arguably because there was already little conflict between the LF outputs. It should be noted, however, that the results of our Snorkel-based model rely almost exclusively on local information pertaining to the DUs in each candidate pair, although we did add some global constraints with the order of application of our LFs on the data.

In the second part of **Chapter 6** we aimed to improve and build the global structure of the different dialogues by implementing a decoding step. A set of highly accurate predictions for individual candidates does not necessarily lead to accurate discourse structures; for instance, without global structural constraints, GEN and local models may not yield the directed acyclic graphs (DAGs) required by SDRT. As Afantenos et al. (2015); Perret et al. (2016), we used the Maximum Spanning Tree (MST) algorithm and a variation thereof, an MS-DAG that includes all high-probability incoming relations that do not create cycles, to ensure that the predicted dialogue structures conformed to more general structural principles. Since discourse attachments in general follow an inverse power law (many short-distance attachments and fewer long-distance attachments), we implemented two MST/MS-DAG variants that always choose the shortest relation among multiple high-probability relations (MST/short and MS-DAG/short). A drawback of this post hoc decoding approach is that the decoding step does not have access to the local information of the DUs or to global information about the discourse graph, since it only takes into account the probabilities of the previous model. Furthermore, the constraint of taking short attachments with high probabilities eliminates our chances of finding long distance attachments (we will discuss in section 8.2 a perception to remedy this lack of global structural information).

While the local model of Perret et al. (2016) did not have strong results, they dramatically improved performance by taking the complete graph of their local model and decoding it using the MST algorithm. We improved our local results by about 5 points using the decoding model on the STAC data. The contrast between these model scores is explained by the fact that Perret et al. (2016)'s model only uses local information from the two DUs, while our "local" model actually includes some minimal global structural constraints in addition to local information, so there was less for the decoding step to do.

In **Chapter 7** we qualitatively explored our results by undertaking an error analysis. We wanted to perceive the patterns' constructions that were not expressed in our LFs. Due to time constraints, we used manual annotations of Dialogue Acts and Addresses in our LFs (we published the results in (Badene et al., 2019a,b)). In parallel, our colleagues were able to successfully predict dialogue acts in a corpus of written conversations between tele-advisors and clients using Snorkel (Thompson et al., 2019), so we could have used Snorkel to predict these features as well. We modified our LFs to capture this information syntactically with regular expressions. The tasks of segmentation, attachment and identifying discourse relations are interrelated, and our rules for attachment even rely on information about discourse relations and their context, as explained in Chapter 5. This made it natural to proceed with an error analysis by looking at particular relations to see how we did on attachments of a particular kind. In the chapter, we described experiments that we conducted to improve the precision of our results focusing on the most important relations for meeting transcripts; we reported our results for Question-Answering-Pairs and Acknowledgment relations. The reason why we chose these relations is that accurately modeling Questions-Answering-Pairs and Acknowledgments (and others important relation such as Result and Contrast) is essential for capturing decisions, when the participants' intentions diverge, as well as cases where they are aligned—in other words core components of the game process and arguably core components of a future industrial application that aims to find decisions disagreements and action items within transcripts of business meetings.

To sum up, a central objective of our work (Badene et al., 2019a,b) was to build dialogue models that allow us to exploit the structural and semantic relations that hold between the contents of different utterances in order to automatically extract more detailed information from conversation data. This requires recognizing the role that each statement plays in a given interaction by having access to what was said, to whom it was said, and to the textual and situational context. Our approach, which focused on the prediction of discourse attachments, revolved around weak supervision, or using high-level knowledge in the form of noisy labeling sources to efficiently label massive data-sets of the sort required to train hungry machine learning models. However, our approach of using a local model that is

independent of the global model shows us our limitations in incorporating more complete structural and contextual constraints.

The biggest advantage of the data programming method that we have experienced for this difficult task of predicting discourse structures is the computation of dependency structures for various weak supervision sources without using any labeled data for training. We moved beyond a model-centric approach, where the data is fixed and we iteratively improve the code/model and parameters, to a data-centric view where data consistency is paramount. Snorkel drastically reduces the costs involved with annotations, both economic and time-wise. The annotations are based on LFs, each of which provides distinct useful cues that adapt and contribute to some understanding of the training data. Snorkel provides many functionalities to iterate on the LFs and investigating their performances in coverage, accuracy, conflict, and polarity.

8.2 Perspectives

The work presented here evokes many other questions for future study. One such question is how the postulated representations should be further formalized, and how reasoning with these formalizations is to be performed. A second question is how this conception of discourse processing may be integrated with theories of discourse structure. While we have looked primarily at two-clause structures, the ramifications that the claims have on multi-clause discourse structure require further investigation. Such studies will form the basis for further characterization of the role of coherence establishment in discourse analysis.

8.2.1 Enhancement

As we have discussed above, a major limitation of our approach is the use of a local model that is independent of the decoding model. We need to draw on the formalisms described in Chapter 2 and be able to incorporate more complete structural and contextual constraints such as the Right Frontier Constraint (RFC) (Grosz and Sidner, 1986; Polanyi, 1988; Webber, 1988). A possible way to model the global structure of a conversation is to graphically represent the interactions as formalized by SDRT, so as to find the available sites for attaching the next DUs by jointly taking into account the probabilities provided by the Generative Model for the choice of the attachment point and the subordinating/coordinating type of LF that attached the pair of DUs. In this way the arcs of the DAG have two items of information; the probability of attachment and the type of subordinating or coordinating link. Inspired by the work of Shi and Huang (2019), we want to find a way to connect our decoding process to

the discourse relations pattern and incorporate more sophisticated global constraints. In the same way, we want to incrementally build our dialogue structures.

For the linguistic aspect of the data, we did not include important linguistic cues for discourse structure in our LFs. Examples of such cues are temporal adverbials and changes in verbal tense and aspect; these have driven the discussion of discourse structure in the linguistic literature—for instance, (Altshuler, 2014, 2016; Asher and Guéron, 2015; Lascarides and Asher, 1993). If our LFs incorporate some expressions that include verbal forms, we need to construct patterns to encompass verbal tenses and their aspects.

At the time of finalizing this dissertation, Snorkel team released the AI platform[1] which uses data-driven programming to easily annotate data. We have yet to study the new features of this platform. Nevertheless, previous versions of Snorkel can also be easily upgraded. The Gibbs sampling used in the original generative model we have used could be replaced by more sophisticated sampling approaches that can perform better, such as Swendsen-Wang sampler used to sample from any graphical model (see Chapter 4). Moreover, if the concept of the generative model is a successful option for classification, other ways of inferring more complex structures should be explored.

The industrial applications we want to develop for spoken conversations (discussed in the next section) pose several challenges that we need to address before transferring our data programming approach for inferring discourse structures for dialogues.

8.2.2 Applications

With the development of digital technologies, many new uses are emerging: on the one hand, smartphones, tablets and connected objects create a pervasive environment where humans are immersed in a continuous flow of information that they need to process, and on the other, the need for customized collaborative assistance between multiple users is more and more pressing. We are interested in the applications of these technologies in the area of professional collaboration, a field in which discourse understanding is highly expected. We are particularly interested in organizing and exploiting information related to meetings but also information conveyed via platforms for instant messaging (such as Slack or Linagora's platform Twake[2]). Finding ways to formally model the understanding of linguistic interactions in relation to the context of different interlocutors (their commitments, their availability, unoccupied meeting rooms, shared documents etc.) will be a huge boon for these industrial applications.

[1] https://snorkel.ai
[2] https://twake.app

As a part of the LinTO project[3], which supported the work in this thesis, the Linagora research team began the design of its Conversation Manager (CM), a tool for editing transcripts and producing summaries, as a complement to its LinTO platform, which provides a variety of voice-operated solutions for businesses. The CM has two main components. The Transcript Editor is a transcription tool that allows users to read and edit the transcript of a conversation, and will soon allow them to add manual and automatic annotations to highlight important points such as action items, decisions, dates, questions and their answers, and so on. The second component, the Summary Assistant, allows users to exploit transcripts and annotations from the Transcript Editor to produce summaries and meeting minutes. The LinTO project involved several research areas: automatic recognition of command and conversational speech, Speaker diarization, natural language processing and understanding, discourse analysis, text summary and discourse structure.

The objective of our work has been to be able to predict the structure of the conversations in order to provide the necessary elements for summaries, recommendations and other modules relevant to the CM, as well as other applications with a high and growing need for accurate discourse structures (Feng et al., 2020; Huber and Carenini, 2020a; Ji and Smith, 2017; Liu et al., 2019; Nejat et al., 2017). We believe that data-driven programming is a powerful way to integrate different types of information (audio, video, calendar, textual content) to develop various applications in this context. Many questions arise regarding the representation of oral conversations in our applications (overlaps or false starts) but also regarding the editing and deletion of messages in collaborative chat platforms that we need to address. While research on discourse analysis was largely limited to English in the LinTO project due to a lack of corpora for multi-party conversations in French, the recently begun ANR project SUMM-RE (ANR-20-CE23-0017) will allow us to extend our weak supervision model to treat meeting-style conversations in French, as part of the project involves creating such a corpus.

[3]https://linto.ai/

Bibliography

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169.

Afantenos, S., Asher, N., Benamara, F., Bras, M., Fabre, C., Ho-Dac, L.-M., Le Draoulec, A., Muller, P., Péry-Woodley, M.-P., Prévot, L., et al. (2012a). An empirical resource for discovering cognitive principles of discourse organisation: the annodis corpus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA).

Afantenos, S., Asher, N., Benamara, F., Cadilhac, A., Dégremont, C., Denis, P., Guhe, M., Keizer, S., Lascarides, A., Lemon, O., et al. (2012b). Modelling strategic conversation: model, annotation design and corpus. In *Proceedings of the 16th Workshop on the Semantics and Pragmatics of Dialogue (Seinedial), Paris*.

Afantenos, S., Kow, E., Asher, N., and Perret, J. (2015). Discourse parsing for multi-party chat dialogues. In *Association for Computational Linguistics (ACL)*.

Alex, B., Nissim, M., and Grover, C. (2006). The impact of annotation on the performance of protein tagging in biomedical text. In *LREC*, pages 595–600.

Altshuler, D. (2014). Discourse transparency and the meaning of temporal locating adverbs. *Natural Language Semantics*, 22(1):55–88.

Altshuler, D. (2016). *Events, states and times: An essay on narrative discourse in English*. Walter de Gruyter GmbH & Co KG.

Angelidis, S. and Lapata, M. (2018). Multiple instance learning networks for fine-grained sentiment analysis. *Transactions of the Association for Computational Linguistics*, 6:17–31.

Asher, N. (1993). *Reference to abstract objects in discourse*, volume 50. Springer Science & Business Media.

Asher, N. (2000). Discourse structure and the logic of conversation. In *Current Research in the Semantics Pragmatics Interface*, pages 1–28. Elsevier.

Asher, N. and Guéron, J. (2015). Perfect Puzzles in Discourse. In Guéron, J., editor, *Sentence and Discourse*, volume chapter 8. Oxford University Press.

Asher, N., Hunter, J., Morey, M., Benamara, F., and Afantenos, S. D. (2016). Discourse structure and dialogue acts in multiparty dialogue: the stac corpus. In *LREC*.

Asher, N., Hunter, J., and Thompson, K. (2020). Modelling structures for situated discourse. *Dialogue & Discourse*, 11(1):89–121.

Asher, N. and Lascarides, A. (1998). Questions in dialogue. *Linguistics and philosophy*, 21(3):237–309.

Asher, N. and Lascarides, A. (2003). *Logics of conversation*. Cambridge University Press.

Asher, N., Muller, P., and Afantenos, S. (2011). Complex discourse units and their semantics. *nation*, 2:7.

Asher, N. and Vieu, L. (2005). Subordinating and coordinating discourse relations. *Lingua*, 115(4):591–610.

Attari, N., Heckmann, M., and Schlangen, D. (2019). From explainability to explanation: Using a dialogue setting to elicit annotations with justifications. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 331–335.

Augenstein, I., Vlachos, A., and Maynard, D. (2015). Extracting relations between non-standard entities using distant supervision and imitation learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 747–757. Association for Computational Linguistics.

Aumayr, E., Chan, J., and Hayes, C. (2011). Reconstruction of threaded conversations in online discussion forums. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 5.

Bach, S. H., He, B., Ratner, A., and Ré, C. (2017). Learning the structure of generative models without labeled data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 273–282. JMLR. org.

Badene, S., Thompson, K., Lorré, J.-P., and Asher, N. (2019a). Data programming for learning discourse structure. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 640–645, Florence, Italy. Association for Computational Linguistics.

Badene, S., Thompson, K., Lorré, J.-P., and Asher, N. (2019b). Weak supervision for learning discourse structure. 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 2296–2305, Hong Kong, China. Association for Computational Linguistics.

Baldridge, J., Asher, N., and Hunter, J. (2007). Annotation for and robust parsing of discourse structure on unrestricted texts. *Zeitschrift für Sprachwissenschaft*, 26(2):213–239.

Barker, C. and Shan, C.-c. (2008). Donkey anaphora is in-scope binding. *Semantics and Pragmatics*, 1:1–1.

Beeri, C., Fagin, R., Maier, D., and Yannakakis, M. (1983). On the desirability of acyclic database schemes. *Journal of the ACM (JACM)*, 30(3):479–513.

Braud, C., Coavoux, M., and Søgaard, A. (2017a). Cross-lingual rst discourse parsing. *arXiv preprint arXiv:1701.02946*.

Braud, C., Lacroix, O., and Søgaard, A. (2017b). Does syntax help discourse segmentation? not so much. In *Conference on Empirical Methods in Natural Language Processing*, pages 2432–2442.

Cao, S., da Cunha, I., and Iruskieta, M. (2016). A corpus-based approach for spanish-chinese language learning. In *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016)*, pages 97–106.

Cao, S., Da-Cunha, I., and Iruskieta, M. (2017a). Toward the elaboration of a spanish-chinese parallel annotated corpus. *EPiC Series in Language and Linguistics*, 2:315–324.

Cao, S., da Cunha, I., and Iruskieta, M. (2018). The rst spanish-chinese treebank. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 156–166.

Cao, S., Xue, N., da Cunha, I., Iruskieta, M., and Wang, C. (2017b). Discourse segmentation for building a rst chinese treebank. In *Proceedings of the 6th Workshop on Recent Advances in RST and Related Formalisms*, pages 73–81.

Cardoso, P. C., Maziero, E. G., Jorge, M. L. C., Seno, E. M., Di Felippo, A., Rino, L. H. M., Nunes, M. d. G. V., and Pardo, T. A. (2011). Cstnews-a discourse-annotated corpus for single and multi-document summarization of news texts in brazilian portuguese. In *Proceedings of the 3rd RST Brazilian Meeting*, pages 88–105.

Carenini, G., Ng, R. T., and Zhou, X. (2007). Summarizing email conversations with clue words. In *Proceedings of the 16th international conference on World Wide Web*, pages 91–100.

Carlson, L. (1984). *Well in dialogue games*. John Benjamins Publishing Company.

Carlson, L., Marcu, D., and Okurowski, M. E. (2003). Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, pages 85–112. Springer.

Carlson, L., Okurowski, M. E., and Marcu, D. (2002). *RST discourse treebank*. Linguistic Data Consortium, University of Pennsylvania.

Caruana, R. (1998). Multitask learning: A knowledge-based source of inductive bias, ser. *Learning to Learn. US: Springer*.

Chapelle, O., Scholkopf, B., and Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Chattopadhyay, P., Yadav, D., Prabhu, V., Chandrasekaran, A., Das, A., Lee, S., Batra, D., and Parikh, D. (2017). Evaluating visual conversational agents via cooperative human-ai games. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 5.

Chib, S. and Greenberg, E. (1995). Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335.

Chomsky, N. (1993). *Lectures on government and binding: The Pisa lectures*. Number 9. Walter de Gruyter.

Chu, Y.-J. (1965). On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

Da Cunha, I., Torres-Moreno, J.-M., and Sierra, G. (2011). On the development of the rst spanish treebank. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 1–10.

Dalrymple, M. (2005). Against reconstruction in ellipsis. In *Ellipsis and nonsentential speech*, pages 31–55. Springer.

Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J. M., Parikh, D., and Batra, D. (2017). Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 326–335.

Das, D., Scheffler, T., Bourgonje, P., and Stede, M. (2018). Constructing a lexicon of English discourse connectives. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 360–365, Melbourne, Australia. Association for Computational Linguistics.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the emalgorithm. *Journal of the Royal statistical Society*, 39(1):1–38.

Denis, P. and Muller, P. (2011). Predicting globally-coherent temporal structures from texts via endpoint inference and graph decomposition. In *IJCAI-11-International Joint Conference on Artificial Intelligence*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Druck, G., Settles, B., and McCallum, A. (2009). Active learning by labeling features. In *Proceedings of the 2009 conference on Empirical methods in natural language processing*, pages 81–90.

Duverle, D. A. and Prendinger, H. (2009). A novel discourse parser based on support vector machine classification. 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: Volume 2-Volume 2*, pages 665–673. Association for Computational Linguistics.

Edmonds, J. (1967). Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Elsner, M. and Charniak, E. (2010). Disentangling chat. *Computational Linguistics*, 36(3):389–409.

Elsner, M. and Charniak, E. (2011). Disentangling chat with local coherence models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1179–1189.

Feng, V. W. and Hirst, G. (2014a). A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521.

Feng, V. W. and Hirst, G. (2014b). Two-pass discourse segmentation with pairing and global features. *arXiv preprint arXiv:1407.8215*.

Feng, X., Feng, X., Qin, B., Geng, X., and Liu, T. (2020). Dialogue discourse-aware graph convolutional networks for abstractive meeting summarization. *arXiv preprint arXiv:2012.03502*.

Fernández, R., Ginzburg, J., and Lappin, S. (2007). Classifying non-sentential utterances in dialogue: A machine learning approach. *Computational Linguistics*, 33(3):397–427.

Ferracane, E., Durrett, G., Li, J. J., and Erk, K. (2019). Evaluating discourse in structured text representations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 646–653, Florence, Italy. Association for Computational Linguistics.

Fisher, S. and Roark, B. (2007). The utility of parse-derived features for automatic discourse segmentation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 488–495.

Geach, P. T. (1962). *Reference and generality*. Cornell University Press.

Geiger, D., Verma, T., and Pearl, J. (1990). d-separation: From theorems to algorithms. In *Machine Intelligence and Pattern Recognition*, volume 10, pages 139–148. Elsevier.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741.

Geurts, B., Beaver, D. I., and Maier, E. (2020). Discourse Representation Theory. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Spring 2020 edition.

Ginzburg, J. (2012). *The Interactive Stance*. Oxford University Press.

Ginzburg, J. (2016). The semantics of dialogue.

Groenendijk, J. and Stokhof, M. (1990). Dynamic montague grammar.

Grosz, B. J. and Sidner, C. L. (1986). Attention, intentions, and the structure of discourse. *Computational linguistics*, 12(3):175–204.

Guz, G. and Carenini, G. (2020). Coreference for discourse parsing: A neural approach. In *Proceedings of the First Workshop on Computational Approaches to Discourse*, pages 160–167.

Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947.

Hayashi, M. (2012). *Turn Allocation and Turn Sharing*, pages 167–190. John Wiley & Sons, Ltd.

Heim, I. (1990). E-type pronouns and donkey anaphora. *Linguistics and philosophy*, pages 137–177.

Heim, I. R. (1982). *The semantics of definite and indefinite noun phrases*. University of Massachusetts Amherst.

Hernault, H., Prendinger, H., du Verle, D. A., and Ishizuka, M. (2010). Hilda: A discourse parser using support vector machine classification. *Dialogue & Discourse*, 1(3):1–33.

Hobbs, J. R. (1979). Coherence and coreference. *Cognitive science*, 3(1):67–90.

Hobbs, J. R. (1985). On the coherence and structure of discourse. *Report No. CSLI-85-7, Center for the Study of Language and Information*.

Huber, P. and Carenini, G. (2019). Predicting discourse structure using distant supervision from sentiment. 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 2306–2316, Hong Kong, China. Association for Computational Linguistics.

Huber, P. and Carenini, G. (2020a). From sentiment annotations to sentiment prediction through discourse augmentation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 185–197, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Huber, P. and Carenini, G. (2020b). MEGA RST discourse treebanks with structure and nuclearity from scalable distant sentiment supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7442–7457, Online. Association for Computational Linguistics.

Hunter, J., Asher, N., and Lascarides, A. (2015). Integrating non-linguistic events into discourse structure. In *Proceedings of the 11th international conference on computational semantics*, pages 184–194.

Hunter, J., Asher, N., and Lascarides, A. (2018). A formal semantics for situated conversation. *Semantics and Pragmatics*, 11. DOI: http://dx.doi.org/10.3765/sp.11.10.

Ilinykh, N., Zarrieß, S., and Schlangen, D. (2019). Meetup! a corpus of joint activity dialogues in a visual environment. *arXiv preprint arXiv:1907.05084*.

Jarrett, K., Kavukcuoglu, K., LeCun, Y., et al. (2009). What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, pages 2146–2153. IEEE.

Jensen, F., Lauritzen, S., and Olesen, K. (1990). Bayesian updating in causal probabilistic networks by local computations.

Ji, Y. and Eisenstein, J. (2014). Representation learning for text-level discourse parsing. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 13–24.

Ji, Y. and Smith, N. (2017). Neural discourse structure for text categorization. *arXiv preprint arXiv:1702.01829*.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1998). An introduction to variational methods for graphical models. In *Learning in graphical models*, pages 105–161. Springer.

Joty, S., Carenini, G., and Lin, C.-Y. (2011). Unsupervised modeling of dialog acts in asynchronous conversations. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1807.

Joty, S., Carenini, G., Ng, R., and Mehdad, Y. (2013a). Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 486–496.

Joty, S., Carenini, G., and Ng, R. T. (2013b). Topic segmentation and labeling in asynchronous conversations. *Journal of Artificial Intelligence Research*, 47:521–573.

Joty, S., Carenini, G., and Ng, R. T. (2015). Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41(3):385–435.

Jurafsky, D. and Martin, J. H. (2014). Speech and language processing. vol. 3. *US: Prentice Hall*.

Kamp, H. (1981). A theory of truth and semantic representation. *Formal semantics-the essential readings*, pages 189–222.

Kamp, H. (2016). Semantics i, ut spring 2016. from montague grammar to drt.

Kamp, H. and Reyle, U. (1993). From discourse to logic; an introduction to modeltheoretic semantics of natural language, formal logic and drt.

Kehler, A. (1993). The effect of establishing coherence in ellipsis and anaphora resolution. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 62–69.

Kehler, A. (1994). Common topics and coherent situations: Interpreting ellipsis in the context of discourse inference. *arXiv preprint cmp-lg/9405010*.

Kim, S. N., Wang, L., and Baldwin, T. (2010). Tagging and linking web forum posts. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 192–202.

King, G. and Zeng, L. (2001). Logistic regression in rare events data. *Political analysis*, 9(2):137–163.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koo, T., Globerson, A., Carreras, X., and Collins, M. (2007). Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150.

Kottur, S., Moura, J. M., Parikh, D., Batra, D., and Rohrbach, M. (2018). Visual coreference resolution in visual dialog using neural module networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 153–169.

Kschischang, F. R., Frey, B. J., and Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519.

Lascarides, A. and Asher, N. (1993). Temporal interpretation, discourse relations and commonsense entailment. *Linguistics and philosophy*, 16(5):437–493.

Lauritzen, S. L. and Spiegelhalter, D. J. (1990). Readings in uncertain reasoning, chapter local computations with probabilities on graphical structures and their application to expert systems.

Li, S., Wang, L., Cao, Z., and Li, W. (2014). Text-level discourse dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25–35. Association for Computational Linguistics.

Liu, Y. and Lapata, M. (2018). Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

Liu, Y., Titov, I., and Lapata, M. (2019). Single document summarization as tree induction. 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 1745–1755.

Loh, P.-L. and Wainwright, M. J. (2013). Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses. *The Annals of Statistics*, pages 3022–3049.

Mann, W. C. and Thompson, S. A. (1987). Rhetorical structure theory: Description and construction of text structures. In *Natural language generation*, pages 85–95. Springer.

Mann, W. C. and Thompson, S. A. (1988). Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Marcu, D. (1996). Building up rhetorical structure trees. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1069–1074.

Marcu, D. (1997). From discourse structures to text summaries. In *Intelligent Scalable Text Summarization*.

Marcu, D. (1998). To build text summaries of high quality, nuclearity is not sufficient. In *Working Notes of the AAAI-98 Spring Symposium on Intelligent Text Summarization*, pages 1–8.

Marcu, D. (2000). *The theory and practice of discourse parsing and summarization*. MIT press.

Matthiessen, C. M. and Teruya, K. (2015). Grammatical realizations of rhetorical relations in different registers. *Word*, 61(3):232–281.

Maziero, E. G., Hirst, G., and Pardo, T. A. S. (2015). Semi-supervised never-ending learning in rhetorical relation identification. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 436–442.

McCowan, I., Carletta, J., Kraaij, W., Ashby, S., Bourban, S., Flynn, M., Guillemot, M., Hain, T., Kadlec, J., Karaiskos, V., et al. (2005). The ami meeting corpus. In *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*, volume 88, page 100. Citeseer.

Meyer, T. and Webber, B. (2013). Implicitation of discourse connectives in (machine) translation. In *Proceedings of the Workshop on Discourse in Machine Translation*, pages 19–26.

Miltsakaki, E., Prasad, R., Joshi, A. K., and Webber, B. L. (2004). The penn discourse treebank. In *LREC*. Citeseer.

Montague, R. (1974). Universal grammar. In Thomason, R. H., editor, *Formal Philosophy: Selected Papers of Richard Montague*, number 222–247. Yale University Press, New Haven, London.

Moore, J. D. and Pollack, M. E. (1992). A problem for rst: The need for multi-level discourse analysis. *Computational linguistics*, 18(4):537–544.

Morey, M., Muller, P., and Asher, N. (2017). How much progress have we made on rst discourse parsing? a replication study of recent results on the rst-dt. In *Conference on Empirical Methods on Natural Language Processing (EMNLP 2017)*, pages pp–1330.

Morey, M., Muller, P., and Asher, N. (2018). A dependency perspective on rst discourse parsing and evaluation. *Computational Linguistics*, pages 198–235.

Morgan, N., Baron, D., Edwards, J., Ellis, D., Gelbart, D., Janin, A., Pfau, T., Shriberg, E., and Stolcke, A. (2001). The meeting project at icsi. In *Proceedings of the first international conference on human language technology research*.

Muller, P., Afantenos, S., Denis, P., and Asher, N. (2012). Constrained decoding for text-level discourse parsing. *Proceedings of COLING 2012*, pages 1883–1900.

Murray, I. and Ghahramani, Z. (2012). Bayesian learning in undirected graphical models: approximate mcmc algorithms. *arXiv preprint arXiv:1207.4134*.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Neal, R. M. (1993). *Probabilistic inference using Markov chain Monte Carlo methods*. Department of Computer Science, University of Toronto Toronto, Ontario, Canada.

Nejat, B., Carenini, G., and Ng, R. (2017). Exploring joint neural model for sentence level discourse parsing and sentiment analysis. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 289–298.

Nielsen, T. D. and Jensen, F. V. (2009). *Bayesian networks and decision graphs*. Springer Science & Business Media.

Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Partee, B. H. (1973). Some structural analogies between tenses and pronouns in english. *The Journal of Philosophy*, 70(18):601–609.

Partes, B. H. (1984). Nominal and temporal anaphora. *Linguistics and philosophy*, 7(3):243–286.

Pearl, J. (1982). *Reverend Bayes on inference engines: A distributed hierarchical approach*. Cognitive Systems Laboratory, School of Engineering and Applied Science ....

Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artificial intelligence*, 29(3):241–288.

Pearl, J. (1988 - 2014). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier.

Pearl, J. and Verma, T. S. (1995). A theory of inferred causation. In *Studies in Logic and the Foundations of Mathematics*, volume 134, pages 789–811. Elsevier.

Perret, J., Afantenos, S., Asher, N., and Morey, M. (2016). Integer linear programming for discourse parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 99–109.

Pisarevskaya, D., Ananyeva, M., Kobozeva, M., Nasedkin, A., Nikiforova, S., Pavlova, I., and Shelepov, A. (2017). Towards building a discourseannotated corpus of russian. In *Komp'juternaja Lingvistika i Intellektual'nye Tehnologii*, pages 201–212.

Pitler, E., Raghupathy, M., Mehta, H., Nenkova, A., Lee, A., and Joshi, A. (2008). Easily identifiable discourse relations. In *Coling 2008: Companion volume: Posters*, pages 87–90, Manchester, UK. Coling 2008 Organizing Committee.

Polanyi, L. (1988). A formal model of the structure of discourse. *Journal of pragmatics*, 12(5-6):601–638.

Polanyi, L., Culy, C., Van Den Berg, M., Thione, G. L., and Ahn, D. (2004). A rule based approach to discourse parsing. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue at HLT-NAACL 2004*.

Prasad, R., Joshi, A., and Webber, B. (2010). Realization of discourse relations by other means: Alternative lexicalizations. In *Coling 2010: Posters*, pages 1023–1031.

Prasad, R., Miltsakaki, E., Dinesh, N., Lee, A., Joshi, A., Robaldo, L., and Webber, B. (2007). The penn discourse treebank 2.0. annotation manual. the pdtb research group.

Prasad, R., Webber, B., and Lee, A. (2018). Discourse annotation in the pdtb: The next generation. In *Proceedings 14th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, pages 87–97.

QIU, M. and JIANG, J. (2013). A latent variable model for viewpoint discovery from threaded forum posts. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'13)*, pages 1031–1040. Citeseer.

Rath, G., Resnick, A., and Savage, T. (1961). The formation of abstracts by the selection of sentences. part i. sentence selection by men and machines. *American Documentation*, 12(2):139–141.

Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., and Ré, C. (2017a). Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282.

Ratner, A., De Sa, C., Wu, S., Selsam, D., and Ré, C. (2016). Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29:3567.

Ratner, A., Hancock, B., Dunnmon, J., Sala, F., Pandey, S., and Ré, C. (2019). Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4763–4771.

Ratner, A., Sa, C. D., Wu, S., Selsam, D., and Ré, C. (2017b). Data programming: Creating large training sets, quickly.

Redeker, G., Berzlánovich, I., Van Der Vliet, N., Bouma, G., and Egg, M. (2012). Multi-layer discourse annotation of a dutch text corpus. *age*, 1:2.

Reese, B., Hunter, J., Asher, N., Denis, P., and Baldridge, J. (2007). Reference manual for the analysis and annotation of rhetorical structure (version 1.0). *URL: http://timeml. org/jamesp/annotation_manual. pdf (last access: 7.9. 2015)*.

Reidsma, D. and Carletta, J. (2008). Reliability measurement without limits. *Computational Linguistics*, 34(3):319–326.

Reitter, D. (2003a). Rhetorical theory in latex with the rst package. *URL: http://www. reitter-it-media. de*.

Reitter, D. (2003b). Simple signals for complex rhetorics: On rhetorical analysis with rich-feature support vector models. In *LDV Forum*, volume 18, pages 38–52.

Sacks, H. (2004). An initial characterization of the organization of speaker turn-taking in conversation. *Pragmatics and beyond new series*, 125:35–42.

Schegloff, E. A., Jefferson, G., and Sacks, H. (1977). The preference for self-correction in the organization of repair in conversation. *Language*, 53(2):361–382.

Schlangen, D. (2003). A coherence-based approach to the interpretation of non-sentential utterances in dialogue.

Schlangen, D. (2005). Towards finding and fixing fragments—using ml to identify non-sentential utterances and their antecedents in multi-party dialogue. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 247–254.

Schlangen, D. (2019). Grounded agreement games: Emphasizing conversational grounding in visual dialogue settings. *arXiv preprint arXiv:1908.11279*.

Schlangen, D., Ilinykh, N., and Zarrieß, S. (2018). Meetup! a task for modelling visual dialogue. In *Proceedings of the 22nd Workshop on the Semantics and Pragmatics of Dialogue - Poster Abstracts*, Aix-en-Provence, France. SEMDIAL.

Shenoy, P. P. and Shafer, G. (2008). Axioms for probability and belief-function propagation. In *Classic works of the dempster-shafer theory of belief functions*, pages 499–528. Springer.

Shi, Z. and Huang, M. (2019). A deep sequential model for discourse parsing on multi-party dialogues. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7007–7014.

Sileo, D., Van De Cruys, T., Pradel, C., and Muller, P. (2019). Mining discourse markers for unsupervised sentence representation learning. 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 3477–3486, Minneapolis, Minnesota. Association for Computational Linguistics.

Skantze, G. (2005). Exploring human error recovery strategies: Implications for spoken dialogue systems. *Speech Communication*, 45(3):325–341.

Smith, A. F. M. (1991). Bayesian computational methods. *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, 337(1647):369–386.

Soricut, R. and Marcu, D. (2003). Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 228–235.

Sporleder, C. and Lapata, M. (2005). Discourse chunking and its application to sentence compression. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 257–264.

Stede, M. and Neumann, A. (2014). Potsdam commentary corpus 2.0: Annotation for discourse research. In *LREC*, pages 925–929.

Subba, R. and Di Eugenio, B. (2007). Automatic discourse segmentation using neural networks. In *Proc. of the 11th Workshop on the Semantics and Pragmatics of Dialogue*, pages 189–190.

Surdeanu, M., Hicks, T., and Valenzuela-Escárcega, M. A. (2015). Two practical rhetorical structure theory parsers. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Demonstrations*, pages 1–5.

Taboada, M. and Mann, W. C. (2006). Rhetorical structure theory: Looking back and moving ahead. *Discourse studies*, 8(3):423–459.

Thompson, K., Asher, N., Muller, P., and Auguste, J. (2019). Analyse faiblement supervisée de conversation en actes de dialogue. In *Conférence sur le Traitement Automatique des Langues Naturelles (TALN-PFIA 2019)*, volume 2, pages 159–166. ATALA.

Toldova, S., Pisarevskaya, D., Ananyeva, M., Kobozeva, M., Nasedkin, A., Nikiforova, S., Pavlova, I., and Shelepov, A. (2017). Rhetorical relations markers in russian rst treebank. In *Proceedings of the 6th Workshop on Recent Advances in RST and Related Formalisms*, pages 29–33.

Varma, P., Sala, F., He, A., Ratner, A., and Ré, C. (2019). Learning dependency structures for weak supervision models. In *International Conference on Machine Learning*, pages 6418–6427. PMLR.

Venant, A. (2016). *Structures, commitments and games in strategic conversations*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier.

Venant, A., Asher, N., Muller, P., Denis, P., and Afantenos, S. (2013). Expressivity and comparison of models of discourse structure. Association for Computational Linguistics (ACL).

Walton, D. and Macagno, F. (2017). Profiles of dialogue for relevance. *Informal Logic*, 36(4):523–556.

Wang, H., Wang, C., Zhai, C., and Han, J. (2011a). Learning online discussion structures by conditional random fields. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 435–444.

Wang, L., Kim, S. N., and Baldwin, T. (2010). Thread-level analysis over technical user forum data. In *Proceedings of the Australasian Language Technology Association Workshop 2010*, pages 27–31.

Wang, L., Lui, M., Kim, S. N., Nivre, J., and Baldwin, T. (2011b). Predicting thread discourse structure over technical web forums. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 13–25.

Wang, Y., Yao, Q., Kwok, J. T., and Ni, L. M. (2020). Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34.

Webber, B. (2016). Concurrent discourse relations. In *Computational linguistics and intellectual technologies: Conference (International)"Dialogue 2016" Proceedings.—Moscow. http://www. dialog-21. ru/media/3488/webber. pdf*.

Webber, B. L. (1988). Discourse deixis and discourse processing.

Xue, N., Ng, H. T., Pradhan, S., Rutherford, A., Webber, B., Wang, C., and Wang, H. (2016). Conll 2016 shared task on multilingual shallow discourse parsing. In *Proceedings of the CoNLL-16 shared task*, pages 1–19.

Yoshida, Y., Suzuki, J., Hirao, T., and Nagata, M. (2014). Dependency-based discourse parser for single-document summarization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1834–1839.

Zeldes, A. (2017). The gum corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Zhou, K., Li, A., Yin, Z., and Zong, C. (2010). Casia-cassil: a chinese telephone conversation corpus in real scenarios with multi-leveled annotation. In *LREC*.

Zhou, Y., Lu, J., Zhang, J., and Xue, N. (2014). Chinese discourse treebank 0.5 ldc2014t21. *Web Download. Philadelphia: Linguistic Data Consortium*.

Zhou, Z.-H. (2018). A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53.