GRAPH ALGORITHMS AND VISUALIZATION OF COMPLEX LEGAL CONTRACTS

Jacob Beckerman

A THESIS

in

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Master of Science in Engineering

2020

Supervisor of Thesis

Chris Callison-Burch

Reader of Thesis

Mark Liberman

Graduate Group Chairperson

Swapneel Sheth

GRAPH ALGORITHMS AND VISUALIZATION OF COMPLEX LEGAL CONTRACTS

ACKNOWLEDGEMENT

I would like to thank Professor Chris Callison-Burch for his mentorship and supervision through my undergraduate and graduate life at the University of Pennsylvania, and supervision of this thesis. I would like to thank Professor Mark Liberman for his reading and feedback that helped shape the final version of this work. Finally, I would like to thank Lev Breydo, a Penn Law and Wharton alum, for sharing his knowledge with me on legal aspects that aided in the development of this work.

ABSTRACT

GRAPH ALGORITHMS AND VISUALIZATION OF COMPLEX LEGAL CONTRACTS

Jacob Beckerman

Chris Callison-Burch

Modern legal contracts are structured into modular components. Like computer code, these components can be parsed into a tree, and their composition (through cross-references) forms a directed cyclic graph. Through a novel pre-processing mechanism, this thesis analyses the graphical structure of legal contracts. While all previous work has attempted to analyze the semantics of contracts (e.g. for summarizing, clause extraction, problem-identification) our work asks: **can we accomplish useful tasks from the graph structure alone, without semantic analysis?** As a preliminary work in this new formation of contract analysis, this work is exploratory; our goal simply to explore the graphical patterns through both visual and qualitative methods and glean interesting information about legalese through this new modality. We find that contemporary techniques in graph analysis and visualization yield interesting findings about the structure of legal language. Finally, we present some opportunities for future work in in the subfield.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER 1 : Introduction

1.1. Background

Written contacts are the foundation of commerce as they have allowed humans to codify, persist, and transfer agreements in a timeless manner; much more than could be done verbally. In the rich history of contracting, our focus falls on modern English-language contracts. Through the 18th and 19th century, English-language contracts have developed a common set of characteristics. While these characteristics are non-binding (in other words, there is no common law or otherwise requirement that contracts be written in such a manner), most all contracts executed today tend to follow some certain patterns.

Due to these patterns, the entropy of contracts is lower than that of unstructured text. That is to say, *a posteriori*, once we are told that a given text is a legal contact, we can craft algorithms with the knowledge of the structure embedded in our parsing algorithms; such specialized parsing algorithms will be able to achieve different outcomes then generic parsing such as named entity recognition or dependency parsing.

In this work, we develop a specialized parser for English language contracts, relying on their pseudo-formal structure.

1.2. Contract Formality

In the case of English language contracts, there are a few characteristics worth noting: unlike web or other unstructred text, it can generally be assumed, by-and-large, that contracts will use appropriate capitalization, punctuation, and grammar. Indeed, even when this fails to hold true, it may be desirable that the parsing algorithm

fails gracefully instead of trying to parse the language (as such language may be legally inoperable). Thus, while contracts are constructed from English, the language is rigid enough to form a pseudo-formal language, especially in syntactical elements (described below) but also to some extent in semantics.

The following two subsections analyze the atomic units of legal contracts: sections and defined terms (AKA definitions).

1.2.1. Well-formed Sections

Contacts are structured into sections. Almost always, contracts contain logical headings, which are most often numbered, lettered, or otherwise marked with a monotonically increasing notation. Often, contacts will have subsections, exhibits, schedules, annexes, chapters, parts, etc. Herein, we will refer to all of the aforementioned logical units as "sections", wherein the *type* of section shall be known as the *section literal* and shall take on a specified value, e.g. *literal*=exhibit—schedule—annex—... In addition to the literal, or in the absence thereof, each section will be marked with a monotonically increasing marking herein refereed to as a *section number*.

The *section number* need not be a Arabic numeral, it may take on values such as *1.1, (a), III* etc. Generally, regardless of the particular characteristics of the numbering scheme, it is the case that the author maintains consistency throughout the contact. If it is not the case that the author maintains consistency, we shall call that contract *malformed*; otherwise it is *well-formed*.

In addition to using section breaks to identify an atomic legal proviso (e.g. Governing Law), longer contracts tend to reference sections from other sections. For example, within the body of *Section 7.1* the author might write "pursuant to any event of default in *Section 5.1*". In this case, *Section 5.1* is specifying a list of trigger events, known as events, and *Section 7.1* is specifying some action that will take place if and

Type	Term Name	Definition
Proper noun	Permitted Debt	$1,000,000 or 5x EBITDA
Proper noun	Company	"Company" shall mean ABC, Inc.
Re-definition	Person	"Person" shall mean any natural person, corporation, or trust.
Sense disambiguation	or	"or" shall not be exclusive.

Table 1: Illustration of some common definition types in legal contracts.

when one of those events is to occur. In this manner, a **reference intertwines the semantics of two or more sections**.

The syntax of references, and their detection, can be challenging. Firstly, authors may use a different *section literal* to identify the section. For example, the section may be marked as *Article V* and referenced as *Section 5*, or it may be marked as *5.1* and referred to as *clause 5.1, subsection 5.1, section 5.1, etc.*. We do not consider these references to make the contract malformed; indeed, we try hard to achieve high precision and recall on section references (with both precision and recall of roughly equal importance to our analysis).

1.2.2. Modular Defined Terms

Contracts have defined terms. Almost always, contracts contain defined terms, which are marked through quotation, boldface, underline, italics, parenthesis, or a table. Defined terms have two parts: (1) the name of the term, which is at least one token and generally not more than ten, and (2) the definition of the term, which may be of unbounded length, but generally is up to a few paragraphs in length.

The semantic purpose of these defined terms varies, as illustrated in Table 1. For example, "Permitted Encumberances" definition, taken (in simplied form) from a credit agreement, specifies how much additional debt the borrower may take on. In this case, the definition is not a "standalone" person, place or thing in itself, but rather is a quantity of debt that the second definition "Company" may take on. In traditional Open Information Extraction formalism, (*Company*, *may borrow up to*,

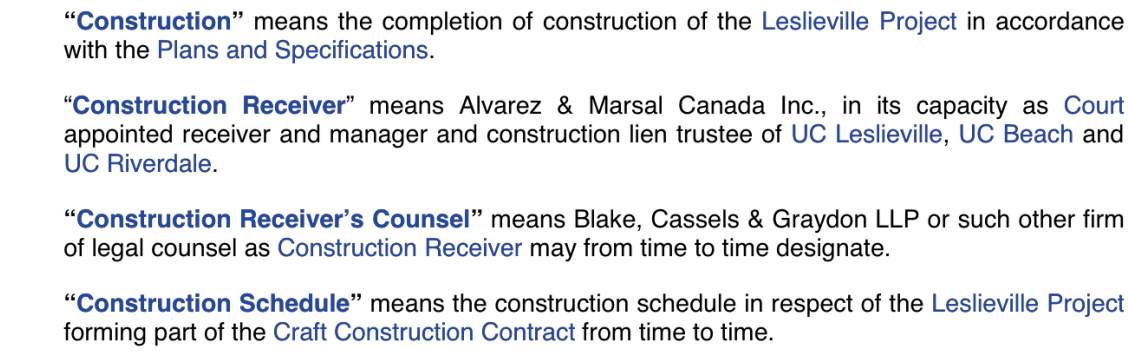

Figure 1: Showing *explicit defined terms* within the *Defined Terms* section in a credit agreement. Screenshot taken from the CoParse viewer.

Permitted Debt). The relation in this tuple, *may borrow up to*, is specified somewhere in the clauses of the document itself, outside of the definition of either "Permitted Debt" and "Company".

Definitions can be specified in contracts in numerous ways; there no no binding statute that governs the format such elements must take on in contracts. In longer contracts, there is often a *Defined Terms* section either at the top of the document or included as a *schedule* or *exhibit* at the bottom of the document, as shown in Figure 1. Herein, we refer to these defined terms as "explicit defined terms" or "explicit definitions".

In both long contracts and short contracts, authors may specify defined terms outside of a section solely decided to that purpose. As shown in Figure 2, the term *Loan Disbursement Account* is defined parenthetically. This parenthetical syntax is used very commonly in legal agreements, and presents challenges for extraction algorithms of the definition body, as it is not clear precisely where the definition should start. Extracting the meaning of such defined terms poses a hard problem that requires knowledge of the semantics of both the definition term and the surrounding context. For example, in this case where the defined term is *Loan Disbursement Account*,

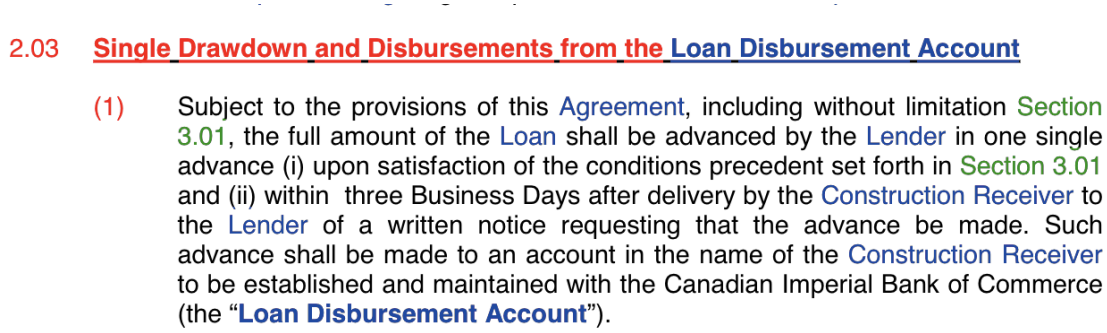

Figure 2: Showing a *inline defined term* called *Loan Disbursement Account* in a credit agreement. Screenshot taken from the CoParse viewer.

our algorithm must understand that an *account* is something relating to a bank and something that has a *in name of* property, in order to properly analyze the proceeding context to establish a reasonable definition.

In our extraction algorithm, due to the complexity this causes, we simply extract the proceeding twenty words from the body of the paragraph, or up to a proceeding defined term, whichever is shorter. In our accuracy study (see Chapter 3), we found this to work well.

1.3. Problem Statement

1.3.1. Motivations

The above sections provided a high-level overview of the syntactical and semantic structure of contracts. While this is interesting, it becomes valuable when paired with a problem to solve; the goal of this section is to outline the purpose of this thesis and give direction to future work.

Contract drafting and review can be a laborious process, causing legal fees that can meaningfully change the economics of doing business (Almeida and Philippon (2007)). These inefficiencies may cause direct costs (the cost of legal counsel) or opportunity costs (lost time and energy). In corporate financial research, "contrac-

tual completeness" and "frictionless contracting" are often assumed (Ganglmair and Wardlaw (2015)). The former states that contracts outline every possible state of the world and specify a remedy, and the second says that the process of drafting, negotiating, and review of contracts is zero (in both money and time terms). In reality, these assumptions are never met, because contacting is not instantaneous, nor is re-negotiation seamless.

For full due-dilligence to be completed, in any time period, we posit that contract will need to be reviewed by *someone* or *something*. If contracts were partially *computable* then more of the review and drafting process could be delegated to a computer rather than a human. This could potentially enable cheaper, faster contracting, thus increasing the pace of business, the velocity of money, and the headache of executives.

While the semantics of language remain allusive from a computability standpoint, this work focuses on **syntactical and grammatical** constructions that could enable a greater share of contract review to be done by a computer. Specifically, we utilize the abstract graphical structure as defined above, together with a parsing system to approximate this structure algorithmically, to produce a graph structure of the contract. In this work we simply intend to analyze this structure to examine the value and limitations of the data contained therein. Our goal is not simply evaluative or inquisitive, but rather to examine the merits of the approach for follow-on work in more high-level analysis tasks:

1. **Extractive summarization of contracts.** Can the graph structure of contracts be used in a summarization algorithm? We posit that if our findings indicate that the graph structure can be used to identify high-importance *sections* and *defined terms* then these sections and defined terms could either (1) be provided as-is as a "tear sheet" summary of the document or (2) composed

in order to form a natural language summary of the document.

2. **Contract completeness.** Can the graph structure of contract be used to determine if a contract is "complete"?

3. **Contract search.** Can we rank search results in a contract based on data from the graph structure? The process of "ctrl-F" through contracts can be laborous; if we could intelligently rank results of sections, definitions, and word matches, both in a single and in many contracts, that could potentially speed up due dilligence as well as contract review.

4. **Contract ergonomics.** While this paper visualises results to answer analytical questions, vizualization may also become a goal in itself. Perhaps there is a way to display contracts to the reviewer in graphical form, or assisted with graphical forms, in order to improve contract review.

1.3.2. Guiding Questions of This Work

Keeping in mind the future applications of this line of research, our goal in this work is to broadly address two questions:

1. **What questions can we answer about the contract?** Is it possible to produce an estimate of the "complexity" of a contract from the graph? Is it possible to identify whether a contract requires additional human review from the graph structure?

2. **What questions can we answer about the clauses and defined terms?** Can we identify important clauses and defined terms from the graph metadata alone, or do we need additional information on semantics. Relatedly, can we produce a summary of the contract from the graph structure alone?

1.4. Document Structure

The structure of this this thesis proceeds as follows: Chapter 1 provided the motivation, introduction and conceptual overview to graphical contract analysis; Chapter 2 frames our task in different lights and reviews the wide array of literature applicable to each framing; Chapter 3 details the extractor system used in this work along with performance benchmarks; Chapter 4 outlines the experimental methodology and code of this thesis; Chapter 5 provides visualizations and a qualitative overview of a small sample of contracts; Chapter 6 analyzes contemporary graph algorithms across our novel graph data structure.

CHAPTER 2 : Related Work

2.1. Introduction

To the author's knowledge, there are no prior works directly relating to graphical analysis of contracts, in the framework provided in Chapter 1. There are several NLP tasks relevant to this work, and several application areas. The rest of this chapter bifurcates related work into "pure" NLP tasks of related work (including those specialized on the legal domain) and works that apply contemporary NLP techniques to the broad area of law and finance.

In this chapter, we bifurcate into application areas and core NLP and review some of the literature in each.

2.2. NLP Tasks

To our knowledge, there is only one prior work of direct similarity, our previous work, Jacob Beckerman (2020). While this work did not produce a graph structure, it laid the foundations for the extractor that we build upon in this work. The goal of Jacob Beckerman (2020) was to produce a document workspace product, as displayed in Figure 3.

A related task to the work is Open Information Extraction, the task of generating n-ary tuples of data. Most often, they are in the form of triplets. This task is often paired with Knowledge Base Construction, which creates entity-relation-entity graphs useful for downstream tasks. For a review of OpenIE and KBC, we refer the reader to Niklaus et al. (2018).

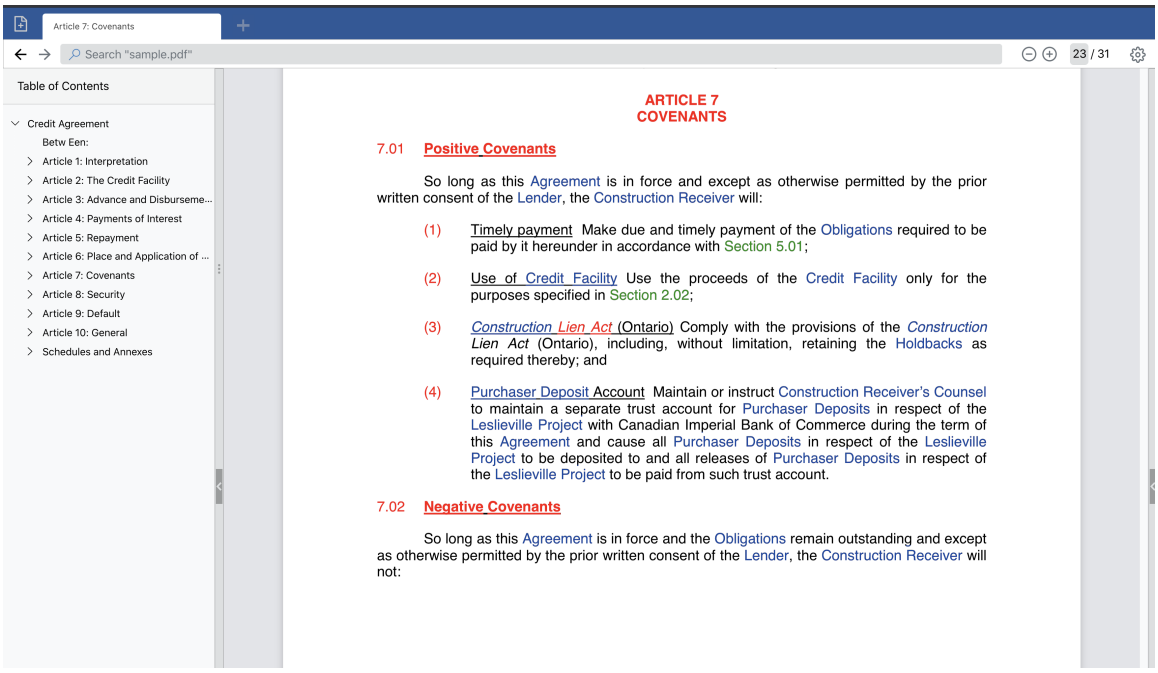

Figure 3: Screenshot of the CoParse viewer, as initially developed in Jacob Beckerman (2020).

2.3. Application Areas

In the literature of finance, there is a varied body of NLP-related research, largely focused on extraction and complexity. Financial research such as Almeida and Philippon (2007) and Ganglmair and Wardlaw (2015) focus on contractual complexity and their effect on the performance of companies. For other works, the focus of the study is not on the contract itself, but rather on data extracted from the financial contract and it's implications on markets (Roberts and Schwert (2020)).

In legal research, the main focus of literature and companies has been on extracting actionable data from contracts. While graph techniques have not previously been used here, other NLP techniques have, such as clustering (Wei et al. (2019)) and conditional random fields (Donnelly and Roegiest (2020)). For a thorough review of NLP in the legal domain, we refer the reader to Bansal et al. (2019).

CHAPTER 3 : Extractor System

3.1. Introduction

The extractor system used in this paper builds upon a previous work by the author (Jacob Beckerman (2020)); thereafter assigned to CoParse, Inc., and is now in use in the Company's eponymous viewer software. The extractor system is largely unchanged in its parsing methodology; indeed, the performance of this system is not the primary focus of this work, rather the analysis of its output is of concern in later chapters. In this work, we significantly overhaul the extractor system of Jacob Beckerman (2020) for increased speed and reliability (the old system frequently crashed on long documents of interest to this study). The following subsections review the architecture of the system, improvements, and benchmarks.

3.2. Previous Work

In Jacob Beckerman (2020), we developed a preliminary system for extracting elements of contracts. This work leaves unchanged the core parsing algorithm, save for some very minor enhancements. We incorporate some relevant definitions (steps in the parsing algorithm) below by reference:

Extract Raw Components: Extracts characters, images, fonts, and vector content from the PDF into memory using the PDFBOX Java library (MIT-license).

Extract Physical Components: Executes "Document Layout Analysis", the combined NLP/Vision problem of reverse-engineering the document structure into characters, words, and text boxes. PDFs don't contain paragraph-data by default: it is simply a collection of vectors rendered at (x,y) positions. We call the output of this step the "Physical DOM". PDFBOX has an internal class to accomplish this step, but

```
▼<term name="Non-Extending Term Lenders" id="191" page="31" y="25.247923">
  ▼<definition>
     <span>“Non-Extending Term Lenders” has the meaning specified in </span>
    ▼<span class="sref" secId="54">
       Section 2.07(d)(ii)
       <br/>
     </span>
   </definition>
  ▼<references>
     <reference page="57" y="76.42291" section="2.07(d)(ii)"/>
     <reference page="57" y="230.11697" section="2.07(d)(ii)"/>
     <reference page="57" y="250.587" section="2.07(d)(ii)"/>
     <reference page="57" y="271.057" section="2.07(d)(ii)"/>
     <reference page="57" y="291.527" section="2.07(d)(ii)"/>
   </references>
 </term>
```

Figure 4: Showing the shape of the *definitions.xml* file; a textual representation of the *ExtractLogical* output.

it doesn't convert text properly (it converts it as glyphs). Thus, we made our own solution using the data in Extract Raw Components, a modified version of PDFBOX's PDFTextStripper (using our own rewrite of the library source code), the rule-based NLP Tokenizer from the NLP4J library to determine word bounds (with our own modifications for legal text), and a lot of our own logic. Our solution is better than any open-source solution we could find to extract text from PDF documents.

Extract Logical Components: in this final step, we process the Physical DOM from Extract Physical Components into a logical tree data structure (the "DOM Tree" or "Logical DOM"). For example, sections can contain sections, terms, images, and paragraphs. This DOM is stored in-memory. This process operates with a sequence of rule-based extractors. For example, since definitions are often enclosed in parenthesis, we have a rule called *bufferedParentheticalExtractor* that extracts this type of syntax. We have a dozen extractors for definitions and several dozen extractors for sections (the section extractors are not well-defined into modular components, rather consisting of a lot of branching logic, so quantification is nebulous).

```
<section title="DEFINITIONS " literal="1" page="4" y="205.06793" id="0" qualified="1" type="ARTICLE">
  <reference page="1" y="62.952877" section="PREAMBLE"/>
  <reference page="1" y="89.59201" section="PREAMBLE"/>
  <section title="Defined Terms" literal="1.01" page="4" y="235.5484" id="1" qualified="1.01" type="SECTION">
    <reference page="1" y="89.20038" section="PREAMBLE"/>
    <reference page="4" y="135.79703" section="PREAMBLE"/>
  </section>
  <section title="Classification of Loans and Borrowings" literal="1.02" page="46" y="294.2429" id="2" qualified="1.02" type="SECTION">
    <reference page="1" y="99.435425" section="PREAMBLE"/>
    <reference page="143" y="67.899506" section="1.01"/>
    <reference page="161" y="67.899506" section="16(c)"/>
  </section>
  <section title="Terms Generally" literal="1.03" page="46" y="344.6829" id="3" qualified="1.03" type="SECTION">
    <reference page="1" y="109.67041" section="PREAMBLE"/>
    <reference page="143" y="78.13449" section="1.01"/>
    <reference page="161" y="78.13449" section="16(c)"/>
  </section>
  <section title="Accounting Terms; GAAP" literal="1.04" page="47" y="65.45288" id="4" qualified="1.04" type="SECTION">
    <reference page="1" y="119.905396" section="PREAMBLE"/>
  </section>
  <section title="Pro Forma Calculations" literal="1.05" page="47" y="197.77289" id="5" qualified="1.05" type="SECTION">
    <reference page="1" y="130.14038" section="PREAMBLE"/>
  </section>
</section>
```

Figure 5: Showing the shape of the *sections.xml* file; a textual representation of the *ExtractLogical* output.

For this work, we output the *ExtractLogical*'s *PDFDOM* subclass to XML, so it can be used in our graph processing in Python. There are two XML files; one for definitions as shown in Figure 4 and another for sections as shown in Figure 5.

In this work, the output of the Logical DOM forms the foundation for the graph visualizations and further processing algorithms. We export the Logical DOM from Java to XML. This XML is later ingested by our Python notebooks, as described in Chapter 4.

Jacob Beckerman (2020) also conducted a performance evaluation of the Logical DOM output with reference to the ground truth as determined by a human annotation effort. As this human annotation effort was not duplicated in this study, and due to the minimal changes to the core algorithm, we produce the metrics from that study in Figures 6, 7, 8. Of note, the accuracy in all aspects (above 94 percent) is good enough that our analysis on the systems graph structure should be above 94 percent. While this could be improved for future work, we believe it sufficient to evaluate the promise of the proposed algorithms.

While the system was accurate on short documents, the speed results as shown in

Summary	Count	Mistakes / Extracted	Standard Deviation	Missing / Actual	Standard Deviation
Basic extraction					
Non-inline definition extraction	1828	0.0%	0.0%	0.0%	0.0%
Top-level section extraction	107	1.9%	5.3%	0.9%	2.9%
2nd-level section extraction	902	4.2%	9.2%	0.0%	0.0%
3rd+ level section extraction	1810	0.2%	0.4%	4.2%	5.9%
Extraction summary	4647	0.9%	5.5%	1.7%	3.7%

Figure 6: Definition and Section Extraction Evaluation Data.

Summary	Count	Mistakes / Extracted	Standard Deviation	Missing / Actual	Standard Deviation
References					
Definition references	2035	2.0%	2.7%	4.1%	5.4%
Section references	2091	2.5%	2.3%	3.5%	2.1%
Reference summary	4126	2.2%	2.5%	3.8%	4.0%

Figure 7: Definition and Section Title Extraction Evaluation Data.

Figure 9 were unacceptable for most applications; while the loading time might be acceptable for shorter documents, the system had a "fat tail" of long-processing times which would cause negative experiences for users (as shown by the outliers in Figure 9). While for this work it could have sufficed to use a computer cluster, or run for days on commodity hardware, in order to pre-process our test set, this would not be sufficient to show practical value. The next section discusses updates made to shorter this processing time and the fat tail.

3.3. Extractor V2: New Updates

In order to process the volume of contracts required for this analysis on commodity hardware, the system needed to be updated for speed. The following sections detail the speed improvements that allowed for the system to operate within sections of before.

Summary	Count	Mistakes / Extracted	Standard Deviation	Missing / Actual	Standard Deviation
Section titles					
Top-level section title extraction	104	0.0%	0.0%	1.9%	4.3%
2nd-level section title extraction	837	1.8%	6.2%	1.2%	2.1%
3rd+ level section title extraction	265	9.4%	21.1%	0.0%	0.0%
Title summary	1206	3.3%	5.6%	1.0%	2.8%

Figure 8: Reference Extraction Evaluation Data.

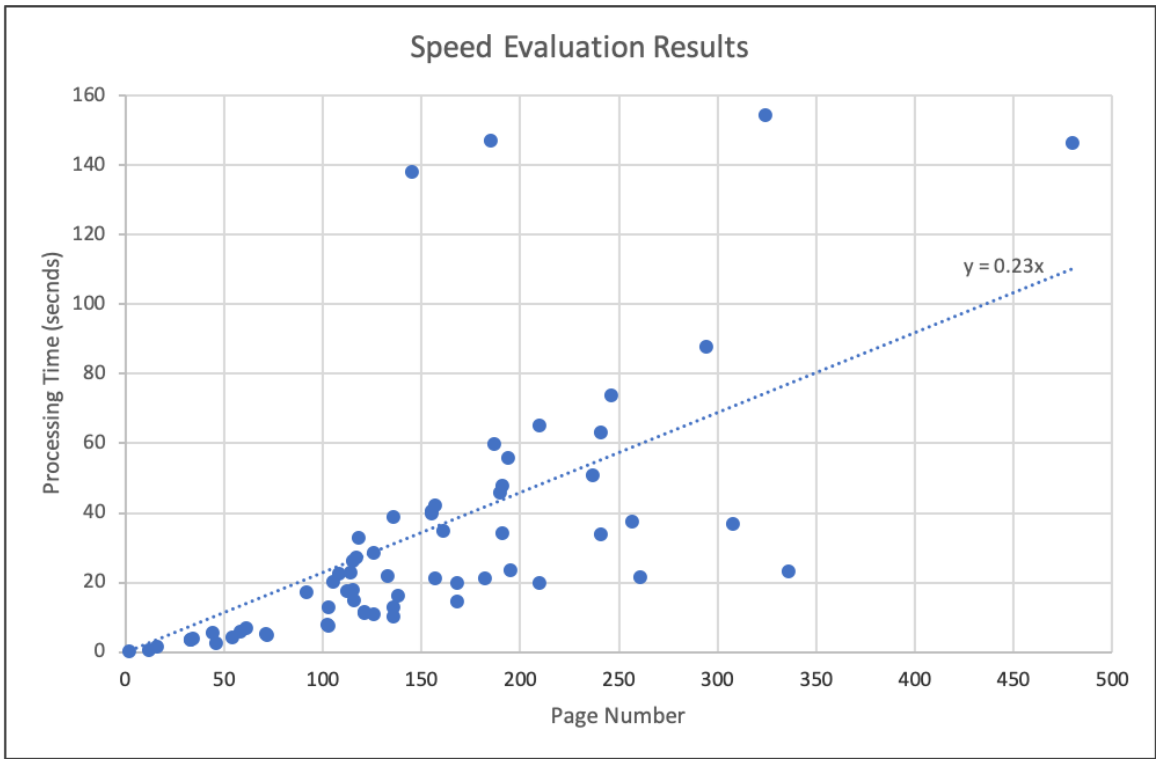

Figure 9: Speed evaluation of the prior work.

3.3.1. JVM pre-warming

The original paper only designed the system for embedded use as a *.jar* file. As such, JVM warming times comprised a significant component of the bench-marked time in the prior work FIgure 9). By embedding a Jetty server and running the application pre-warmed, we saw an average improvement 2.9 seconds off the processing time.

3.3.2. Multi-processing

While the graph construction operation contains many components, Jacob Beckerman (2020) did not make use of any parallelization. In this work, we overhaul the extractor architecture to make use of multiples CPU cores. Walking through Figure 10 from top to bottom: first the *ExtractPhysicalWrapper* makes use of two CPU cores to extract the data from the PDF and perform document layout analysis. One *ExtractPhysical* instantiation, of the same kind as in Jacob Beckerman (2020), operates on even pages while the other operates on odd pages. Critically, once the *ExtractPhysical* component is completed for a given page, the extracted data structure (containing tokens and textboxes for that page) is then passed to the *ExtractLogical* which starts running *up to and including that page*. The *ExtractLogical* process, by it's nature, is not easily parallelizeable: in order to determine the "Logical DOM" we have to serially walk through the document and determine which subsections go under which. Thus, our goal here is to reduce the running time of the algorithm to the time it takes to run *ExtractLogical*, as *ExtractPhysical* can be parallelized. We found that running two *ExtractPhysical* processes was sufficient to ensure the queue to the *ExtractLogical* was non-empty until the document was fully processed, meaning there were no further speedups from further parallelization. Indeed, when using more *ExtractPhysical* cores, we found that multi-threading inefficiencies actually increased the processing time.

All in all, this optimization reduced the additional time for *ExtractPhysical* to practi-

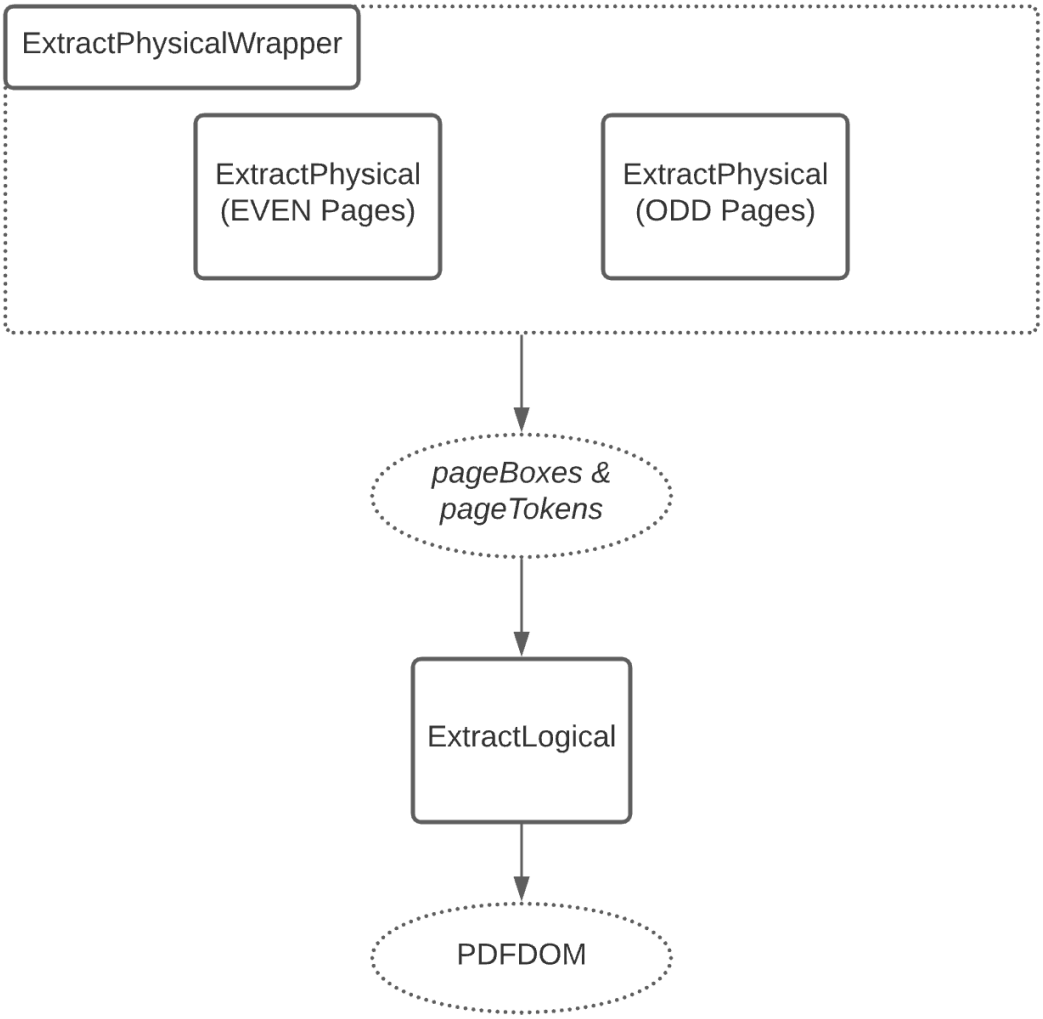

Figure 10: Flow chart of the data flow in the new extractor system architecture.

cally zero on a multi-core (3 or more cores) machine, because it runs asynchronously. This yielded roughly a 2x speedup, as *ExtractPhysical* and *ExtractLogical* took the same amount of time when run serially.

3.3.3. Reference Extraction

As part of *ExtractLogical*, there are two subtasks that require separate passes through the document: (1) constructing the tree structure of the document, including sections and defined terms and (i.e. the document tree) (2) extracting references to these elements to form the directed graph. If sections and terms were always used before they were referenced, then this second pass through the document would not be needed. While the "Defined Terms" section is often at the top of the document, it it legally OK, and common practice to include them in the middle or at the end. Additionally, defined terms may be defined parenthetically, or inline in another manner, anywhere the document. Likewise, sections are commonly referenced before they are seen in the document, reading from from to back, left to right. Thus, it is important to do two passes through the document: first, to complete task (1), and then to complete task (2). The focus of this optimization is on step (2).

In Jacob Beckerman (2020), we implemented a version of the Knuth-Morris-Pratt algorithm as described in Knuth and Pratt (1974). For every term, we would run the algorithm over the document; thus, if there were 200 terms, the algorithm would make 200 passes through the document. Further, since we require that defined terms match in a fuzzy manner, we were running the Porter Stemmer (C.J. van Rijsbergen and Porter (1980)) inside the loop, stemming each word as the algorithm iterated through.

This work optimizes both of these processes. First, each textbox and each defined term is stemmed and cached. Then, the algorithm described in Aho and Corasick

(1975) is used to search for references. The time complexity of the search is $O(n+d+m)$ where n is document length in terms of letters and d is the number of defined terms and m is the number of matches (Aho and Corasick (1975)). This is an improvement over the $O(nd)$ time from Jacob Beckerman (2020). For a long document like the Windstream agreement (depicted graphically in Figure 16), this reduced processing time by an average of 2400ms.

3.3.4. Results

Following the improvements mentioned in the previous sections, we benchmark the optimized version of the system in Figure 11. After the optimizations, the algorithm is roughly 5x faster while maintaining the same accuracy. In a typical workflow, such as reviewing a 200-page credit agreement, this is the difference between waiting ten seconds and fifty seconds, which could drastically improve user-experience in downstream systems that use our methodology as an input.

We believe that there are further optimizations that could be made to the extractor system, without major overhauls to the parsing logic, that would further reduce the processing time. We achieved roughly 10x faster improvement in the algorithm, but due to multi-threading stability bugs (increased average speed but a fat tail of really long processing times), the system was paired back in some places. With more development effort, we believe that speed increases of 10x over the original paper (2x over this paper) are possible, and with architectural changes even more could be achieved.

It should be noted that we also rely on local (commodity) hardware in order to avoid sending the document to the cloud. Both of these experiments were carried out on a 16 inch MacBook Pro, 2020 edition, with 16 GB of RAM and a 6-core Intel processor. With access to cloud computing, these faster results could be achieved, potentially

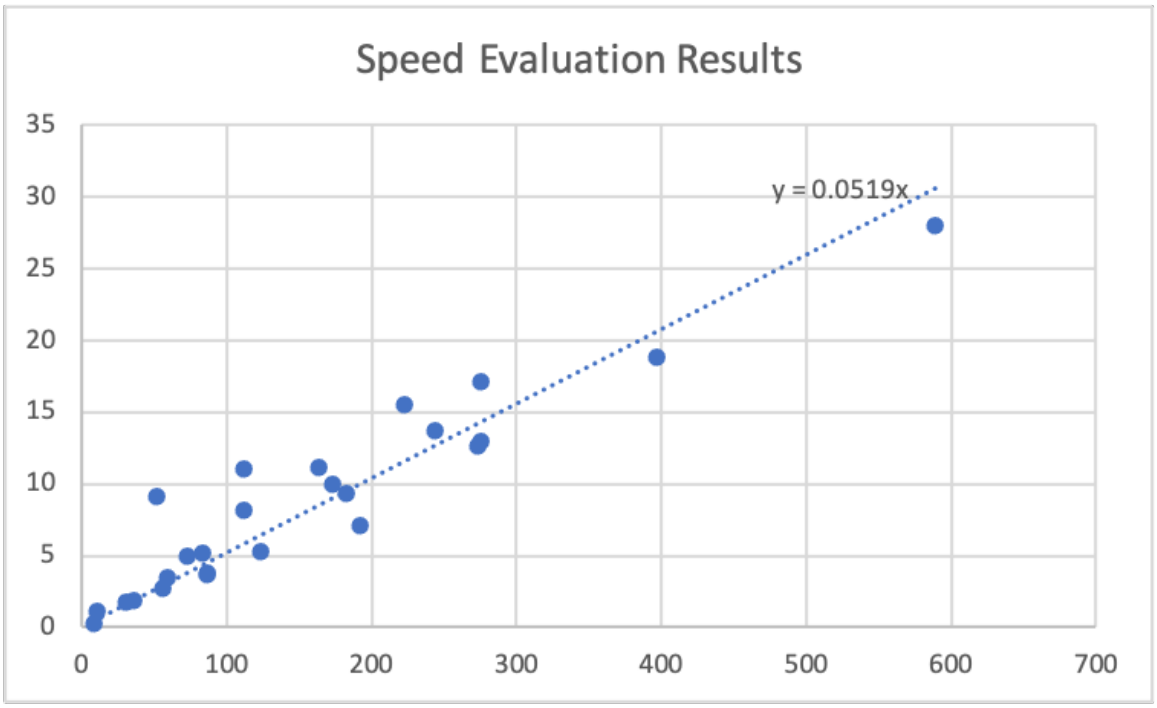

Figure 11: Speed evaluation of overhauled extractor system on a representative set of contracts of varying length.

reducing the processing time down to a negligible amount, meaning downstream consumers of our graph data could be available immediately when the user wants to open and read a document, which could be important for seamless user experience.

CHAPTER 4 : Methodology

4.1. Dataset

The contracts for this work are all in the public domain via the SEC's EDGAR service (www.sec.gov/edgar.shtml). We curate a sample of twenty each of credit agreements, bond indentures, and then a miscellaneous set consisting of an even sample of structured finance, inter-creditor, vendor, merger, acquisition, service, purchasing, licensing, real estate, formation, security, guarantee, disclosure, executive employment, shareholder, reorganization, lease, equity, convertible, and fund formation agreements.

Critically, **we exclude** documents like NDAs, NDIAAs, non-executive employment agreements, order forms, and other shorter contract types. The reason we exlcude these documents, following an initial study, was that these documents don't always contain sections (or numbered sections) and even if they do, they include minimal cross-references. In addition, these documents don't contain an extensive set of defined terms. In other words, the graph is both minimal and small, as shown in Figure 12.

Due to (1) the small number of nodes and (2) the smaller number of references, we chose to exclude these contracts for our test set. A natural questions is: what implication does this have on our results, and the utility of the graph structure for downstream tasks (e.g. extracting summarization)? First, it should be noted that many short contracts NDAs are not heavily negotiated, and are often standardized. However, there are some shorter documents, like employment agreements and letters of intent that may be hotly negotiated. In these cases, while the graph algorithm would provide no utility, they are also short enough to be read in their entirety without

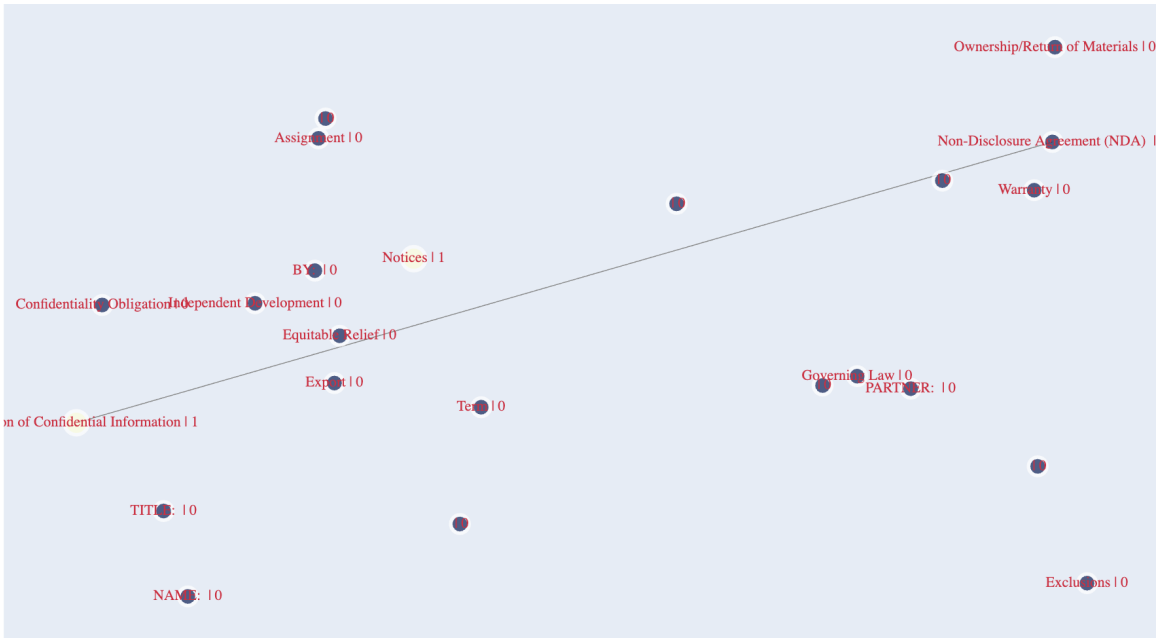

Figure 12: Visualization of a 4-page NDA agreement sourced from EDGAR, showing small sparsely connected graph.

strain. In other words: if the contract is large enough, then it becomes important to have computer-aided summarization, extraction, etc., but if the contract is short then the user can read it themselves. While the sparse analysis of these small graphs may be interesting for future work, we leave them out of our analysis herein.

4.2. Visualization

As an exploratory work, our goal is to investigate the properties of contracts and propose some basic algorithms. We utilize Python's NetworkX (networkx.org) and Plotly (plotly.com) to produce our graphs.

CHAPTER 5 : Visualization and Algorithm Development

The goal of this section is to take advantage of graph visualization techniques to answer the guiding questions set in Chapter 1. **We recommend using Adobe Acrobat to export these figures in this chapter from the PDF, or expanding the PDF and zooming in in order to view the small text**.

This chapter frequently makes use of the word "importance", as per the goals of this work, we are trying to identify important parts of a contract. However, this of course leaves the questions of *importance to whom, and in what context?* For example, a financial analyst may care about the leverage ratios expressed in a credit agreement when trying to decide whether to buy a company's debt in the secondary market. If, a few months later, one of the subsidiaries removes from reach a portion of collateral from the lenders, the same financial analyst may now care about the "guarentors" section. Similarly, a lawyer working on behalf of the Company may care deeply whether the Governing Law is Delaware or New York, while this may not matter to opposing counsel. How do we deal of this problem of *for whom* and *in what context* when deciding what is important? A similar problem arises in summarization research, and while there are opportunities for goal-oriented summarization as in Stewart et al. (2007), we focus on non-goal-oriented. In general, when it comes to contracting, there is a common set of clauses that reasonable readers can agree are generally important. As a guiding rule, these will roughly correspond to the most negotiated pieces, or the ones that would be included in a term sheet prior to the drafting of the full legal agreement.

Since there are innumerable algorithms possible for graphical analysis, we intend to give the reader in this chapter a sense of our experimentation to find the best

algorithms. While we show only a few agreements' graphical representations for clarity, the insights presented and the narrative reflected is representative of our tests on the wider test set, though such visualizations are too long to display in a digestible manner in this work.

5.0.1. First Pass - Sections without Defined Terms

In Figure 13 we begin by visualizing a Canadian Credit Agreement by Alvarez and Marsal. The PDF of this document, augmented with the defined terms and section links can be viewed in the CoParse viewer, gratis, at demo.coparse.com.

We begin by analyzing only the sections and section references, leaving out the definitions. The important sections in this credit agreement (and in credit agreements in general) include the negative covenants section, governing law, and permitted liens/encumbrances/debt sections — these sections specify what the borrower is permitted to do after taking the loan from the lender. Many of the other sections are boilerplate and are skipped over by an adept reader.

In Figure 13, we note a few observations with reference to our goals.

1. **Disconnected sections match our intuition of boilerplate sections**. In the visualization, sections including *Time is of the Essence, General, Interest Calculation of Payments, Amendments and Waivers, General, Maximum Rate of Interest, Schedules, Currency, Conflicts, Place and Application of Payments, Extended Meanings, Address, Positive Covenants* are shown as unconnected nodes in the graph. This matches our intuition for these sections; they are generally boilerplate that is not of primary importance to an analyst.

2. **Connected sections *generally* match our intuition for important sections.** At the same time that disconnected sections are unimportant, some

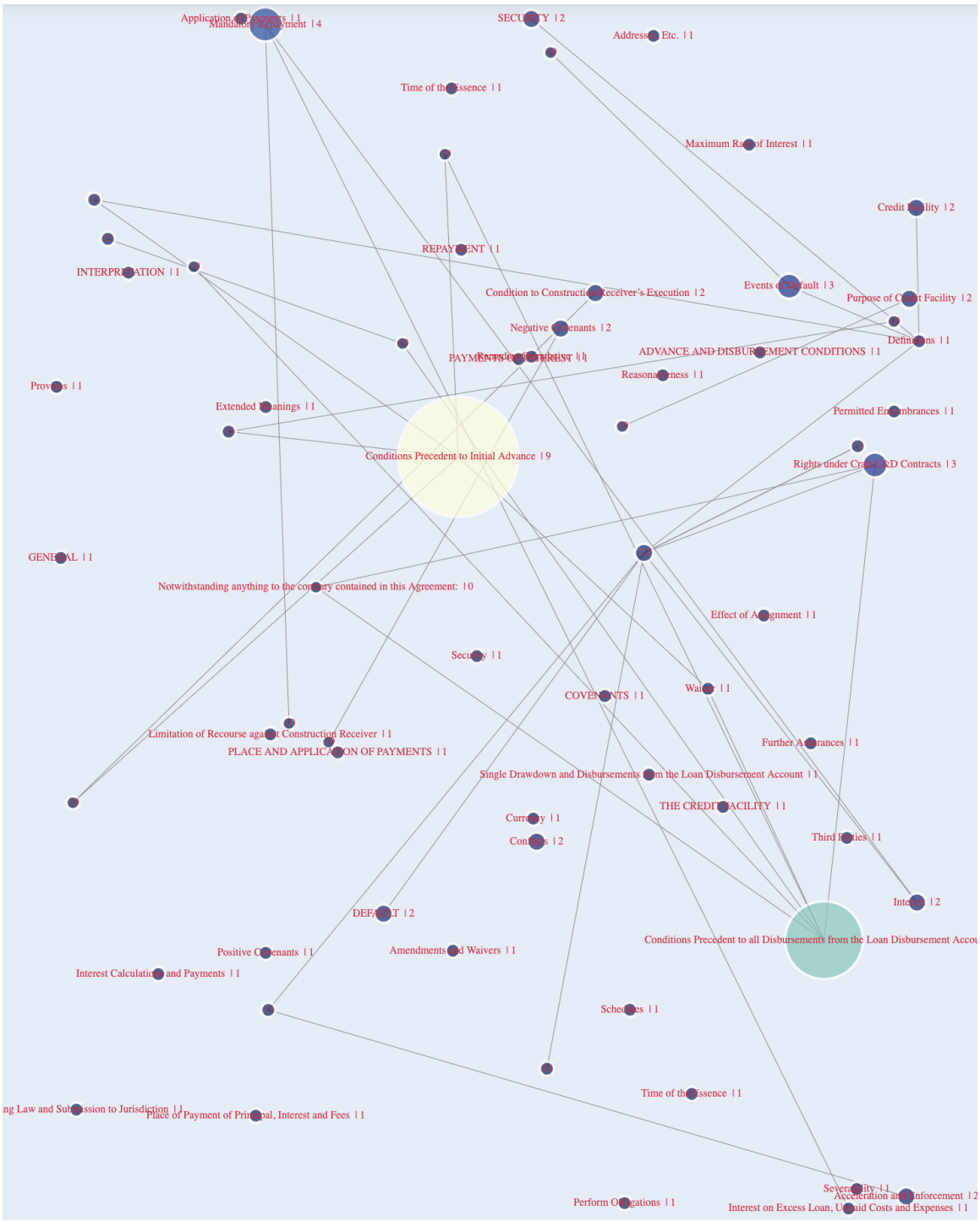

Figure 13: Visualization of a 30-page (short) Canadian Credit Agreement; naive reference algorithm. The size and color of a node indicates section. Reference from the "Table of Contents" page are not shown as edges for visual simplicity, but are included in the count.

key sections such as *Events of Default, Credit Facility, Mandatory Repayment, Rights under Craft CD Contracts* are connected.

3. **Node size does not seem to correlate with importance.** While connected sections seem to correlate with importance, sections such as *Conditions Precedent to Initial Advance* and *Conditions Precedent from the Loan Disbursement Account* are much larger than sections that an adept reader would typically consider more important, such as *Events of Default*.

4. **High-level (i.e. higher in the Table of Contents tree) sections receive fewer references**. This makes sense, as authors often want to be as precise as possible in their drafting, striving for "contractual completeness" as described in Chapter 1.

This preliminary example elicits several important questions as we build on our naive vizualization and processing algorithms, as well as for future work to explore more in depth:

1. **Should definitions and sections operate separately?** In the analysis thus far, the graph structure was limited to sections. When computing the importance-weighted graph, should we consider definitions and sections are disjoint subgraphs? Or should we combine them into one graph for analysis?

2. **Should self-references count as reference?** Oftentimes, as shown in Figure 14, sections reference themselves (or an enclosing parent section) with words to the effect of "as mentioned anywhere in this *Section X.XX*". Such language likely doesn't convey importance as it doesn't link the semantics of multiple sections, it simply references itself in formal terms. In the case of PageRank and other graph processing algorithms, such links are often excluded as they

(2) After the initial advance under Section 3.02(1), the Construction Receiver shall be entitled to disburse amounts from the Loan Disbursement Account upon fulfilment of the conditions in Section 3.02.

Figure 14: Screenshot of the Canadian Credit Agreement from the CoParse Viewer, *Section 3.02(2)* showing showing self-reference to *3.02* and sibling reference to *3.02(1)*.

convey no importance. Herein, we exclude self-references in our algorithm.

3. **Should references from siblings be weighted differently?** As shown in 14, subsections often reference sibling sections i.e. those sections with a common parent in the table of contents tree. These sibling-references are more "local" than references that don't share a parent, so perhaps they don't convey the same sense of importance. One could imagine that instead of breaking up *Section 3.02* into over ten subsections, as done in the Canadian Credit Agreement, the authors could have consolidated them into one large section at the expense of clarity; thus, changing the sibling-references into self-references. However, simply eliminating sibling references will cause issues at higher-in-the-tree sections (for example, at the level of *Articles* in the Canadian Credit Agreement) because these are semantically different sections. Thus, one solution may be to only eliminate sibling references at depth greater than k in the tree. A value of $k = 3$ may seem appropriate for some contracts; however, there are some contracts that use a greater level of granularity in constructing section boundaries, thus ending up with a deeper tree, potentially throwing away semantically meaningful sibling-references. Due to this realized complexity, herein we leave sibling references in our count.

4. **Should references propagate up to parents**? The term "textual body" in this section and afterwards shall have the meaning in the sense of XML,

(11) Terra Firma shall have funded to the Construction Receiver the (a) cost of all Latent Defects discovered by Craft, the Construction Receiver or any other Person relating to the Leslieville Project as of the date the other conditions precedent set out in this Section 3.01 have been satisfied, (b) amount of all "Development Cost Overruns" (as defined in the Craft Development Contract) requested by Craft as of the date the other conditions precedent set out in this Section 3.01 have been satisfied, in each case as required under the TF Cost Overrun Guarantee, and for certainty, inclusive of HST;

Figure 15: Screenshot of the Canadian Credit Agreement from the CoParse Viewer, showing multiple references to *Section 3.02* within the same section.

i.e. *the text directly contained by a section and* ***not*** *including text contained in subsections*. Often, the textual body for non-leaf sections in the table of contents is vacuous; that is to say, these non-leaf sections exist solely to group leaf sections by semantics. Other times, non-leaf sections will include an introduction or conclusion in the textual body, potentially including references to other sections. All in all, we found in our visualizations that the text content tends to be weighted towards the lower nodes in the tree, especially the leaf nodes. This has implications for the estimated importance of non-leaf nodes, since under our naive weighting algorithm depicted in Figure 13 we do not "upstream" references to parents in any manner. Is this an issue? We posit that it depends on the intention of the viewer (if visualization is the end-goal) or the downstream task (in the case of algorithmic end-goal). We will visualize several contracts in next section using both schemes to qualitatively determine which is more meaningful to the user i.e. which most aligns with our abstract notion of importance of various clauses.

5. **Should we use a directed graph or multiple-directed graph representation?** As shown in Figure 15, it is quite often the case that sections will reference other sections more than once. Our naive algorithm has included

multiple references in the count (for clarity, we don't show parallel edges in the figure, but parallel edges are included in the count). If our hypothesis is that semantic importance is conveyed via the link structure, is it the case that multiple references convey additional importance? We will investigate later in this chapter.

Before moving on to algorithmic improvements, we visualize a much longer set of contracts in Figure 16. Again, in this image, we can see that large nodes generally reflect our notions of importance; the key nodes *Leverage Ratio* and *Interest Coverage Ratio*, despite being only two sentence long and specifying a single ratio, are two of the most important sections in the document to an analyst reading the agreement. Other large nodes include *Indebtedness, Events of Default, and Amendments* which are generally important sections.

5.0.2. First Pass - Sections without Defined Terms

Perhaps even more important than sections to the structure of the legal document are defined terms. These modular elements are referenced more frequently than sections. For all unique defined terms, we count a reference as any time that the defined term appears in the document. We also allow matching against stemmed versions of the defined term and body of the document using the algorithm described in C.J. van Rijsbergen and Porter (1980). For example, the term "Loan Parties" will match with "Loan Party" and vise-versa, since both are stemmed to "Loan Part". This stemming seldom results in false matches, as evaluated in Chapter 3 (though one could imagine frequent mis-matches for malicious input documents).

Continuing with the simple reference-weighted algorithm, we visualize the defined term structure in Figure 17. In this graph, size and color represent references from *any* place in the document, and edges are shown when a definition is references from

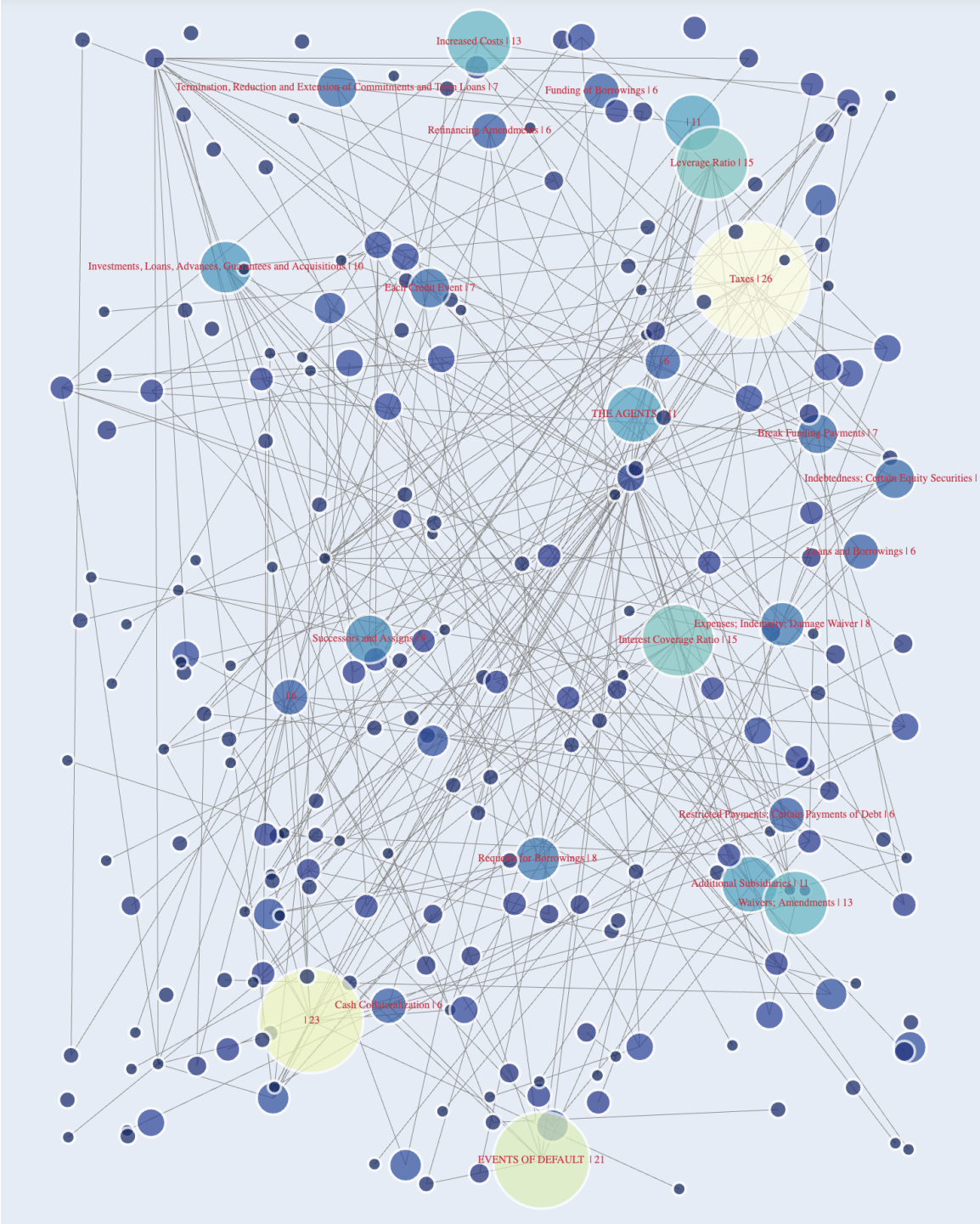

Figure 16: Vizualization of a 275-page (long) Credit, Security, and Guarantee Agreement for Windstream Holdings; naive reference algorithm. Names only displayed for sections with greater than five references.

another definition. The large discrepancy in the size/color of some nodes compared to the number of edges coming out of them (for example, see *Administrative Agent* in the top left) is due to the fact that most references are from the body of the document, not from other defined terms, and thus are not shown in the figure.

Interestingly, in Figure 17, we see the inverse correlation with importance that we saw in Figure 16 of sections:

1. **Larger defined terms represent boilerplate**. For example, *Person, Lenders, Subsidiary, Administrative Agent, Borrower, Collateral Agent* all represent defined terms that are not of highest importance to a financial analyst. This generally matches our intuition, as conversely to sections, boilerplate definitions tend to be used over and over again in the contract. The law firm drafting the contract likely reuse the same contract base across deals, simple redefining these terms, such as "Company" based on the specifics of the deal.

2. **Smaller defined terms include both boilerplate and important definitions**. Since there are only a few larger defined terms in Figure 18, most of the defined terms receive few references, including both important and boilerplate.

Moving on in our algorithmic development and visual explorations, in Figure 18, we limit the scope of the reference-search to definitions. For example, if the definition of "Borrower" is "Company and it's Subsidiaries" there would be two directed edges: *(Borrower, Company)* and *(Borrower, Subsidiaries)*. No longer do we include references not within definition bodies in the size/color of the nodes. For example, if the text in *Section 2.02* reads "the Borrower shall not...", this reference is excluded entirely from Figure 18 as it is not within the body of a definition.

For clarity herein, we refer to the type of reference weighting in Figure 17 as *WDRW*

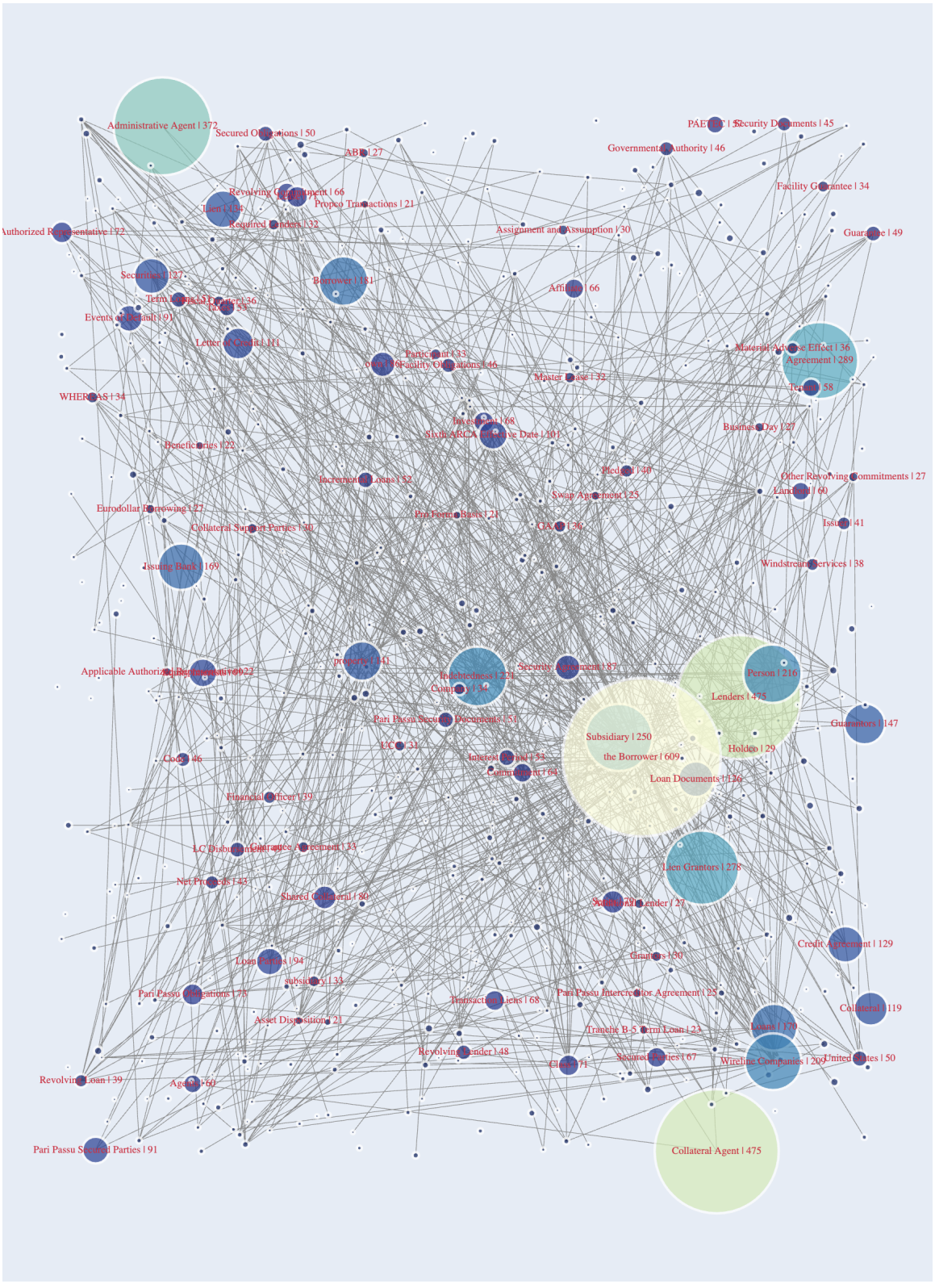

Figure 17: Vizualization of a 275-page (long) Credit, Security, and Guarantee Agreement for Windstream Holdings; visualization of the graph of defined terms and their references using simple reference weighting. Color and size represent the number of references throughout the document.

(whole-document reference weighting) and the type of weighting in Figure 18 as *DORW (definition-only reference weighting).*

The results of Figure 18 are interesting, as they are the direct opposite results of the Figure 17. In this Figure, we find that the more important defined terms to the substance of the credit agreement are the larger ones, for example *Available Distributable Cash, Consolidated Adjusted EBITDA, Collateral and Guarentee Requirement, Available Equity Proceeds, etc.*. Conversely, the large terms in Figure 17 are not even visible on this graph, as they receive less than the 5-reference threshold for showing their title (for visual clarity). Indeed, even without any knowledge of legal language, credit agreements, or finance, any reader can see that the terms of large size in Figure 18 and Figure 17 **are almost completely disjoint.** This is a surprising result: why would it be the case that boilerplate terms are referenced more in the document, while important terms are referenced more in the bodies of other defined terms?

We further explored this quirk on our test set of documents, to examine whether this was idiosyncratic to the Windstream agreement, or was a pervasive property of the reference structure of legal contracts. In Figure 19 we examine the same comparison on an Asset Sale Agreement, a totally different agreement type compared to a credit agreement. For clarity in the smaller figure, we remove edges. In Figure 20, we test the property on the same agreement type as in Figure 17 but in a much longer and more complex document. In Figure 21 we show a project finance agreement (again, a completely separate agreement type). In all three of Figures 19, 20, and 21 we find the same disjointedness property as earlier: WDRW and DORW schemes produce a disjoint set of large nodes. In addition, we find that the finding that WDRW represents boilerplate and DORW represents defined terms holds.

We hypothesize that this pervasive disjointedness is caused by difference in purpose

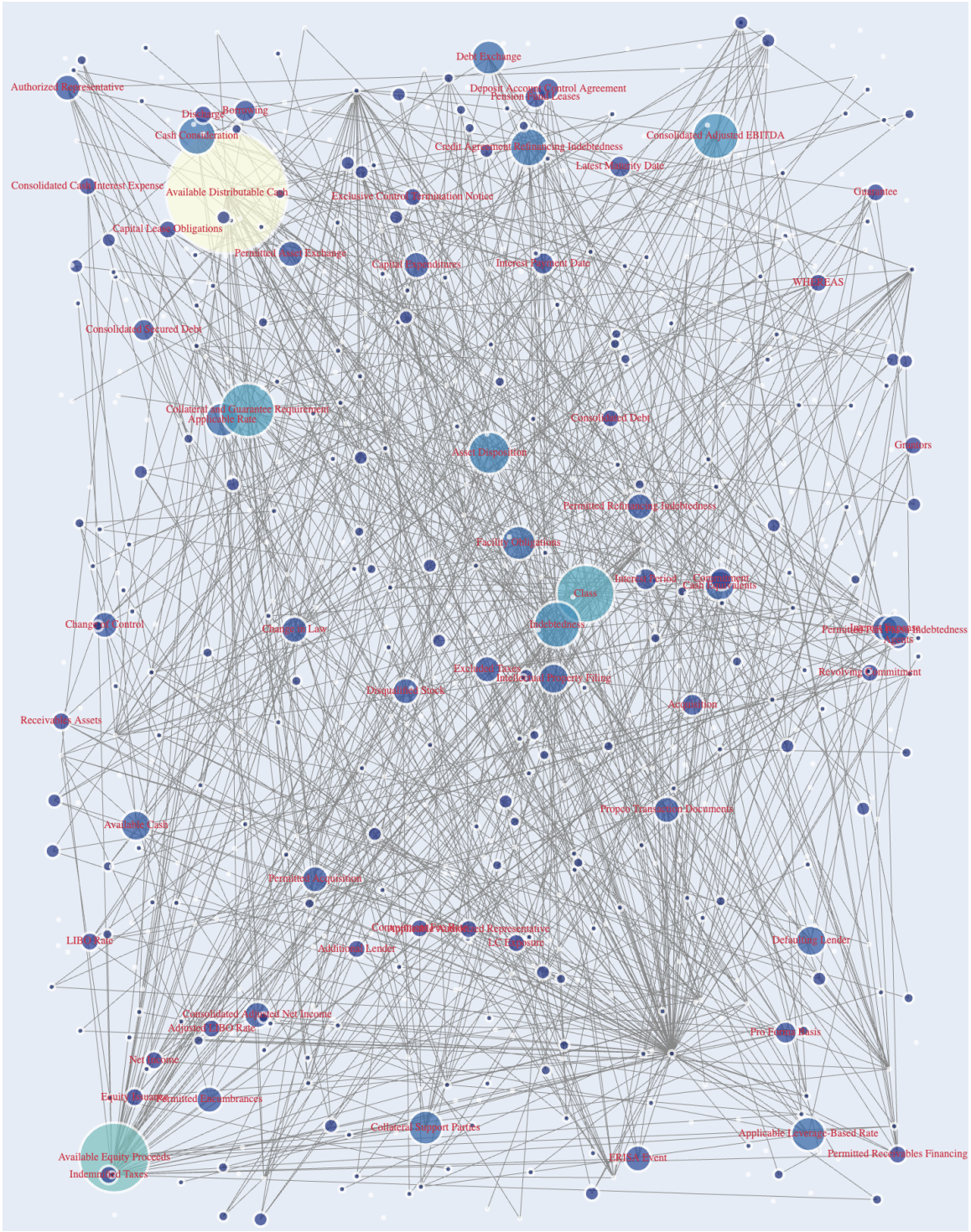

Figure 18: Vizualization of a 275-page (long) Credit, Security, and Guarantee Agreement for Windstream Holding; visualization of the graph of defined terms and their references using simple reference weighting *limiting to those references that are contained in the body of other defined terms.*

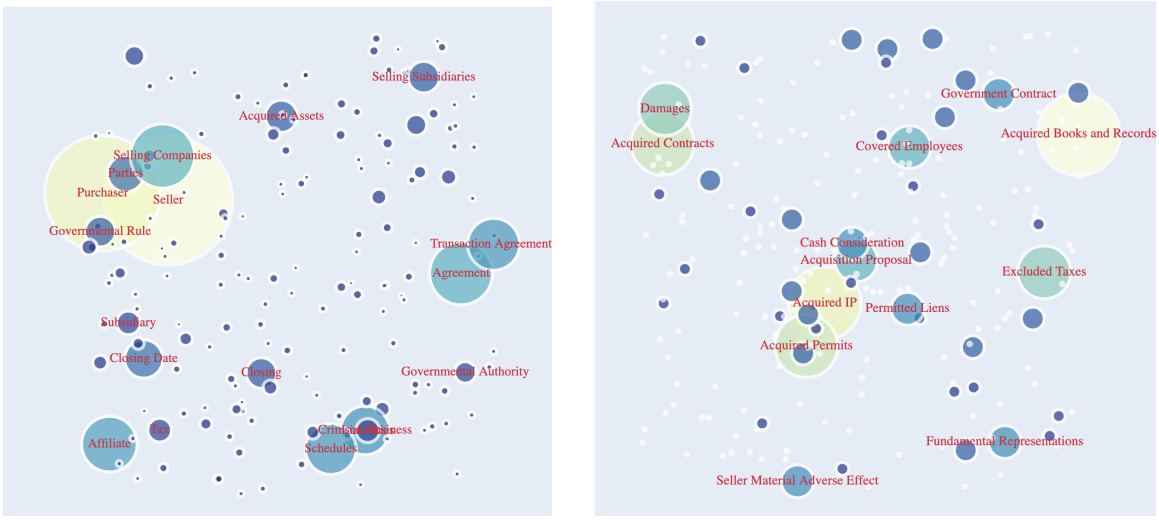

Figure 19: Comparison of WDRW weighting (left) vs. DORW weighting scheme (right) on an Asset Sale Agreement.

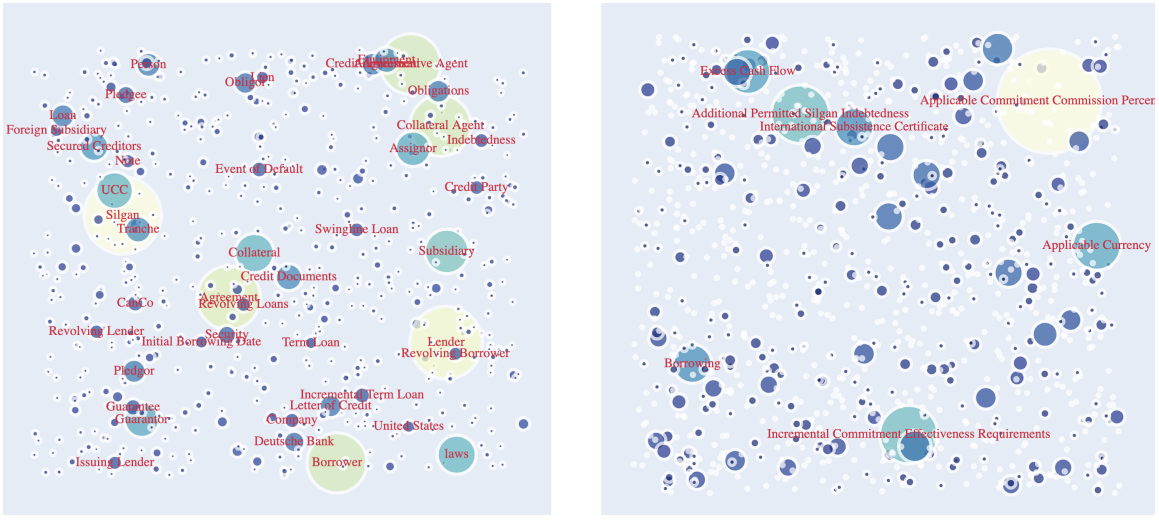

Figure 20: Comparison of WDRW weighting (left) vs. DORW weighting scheme (right) on a very long (600 page) Credit Agreement.

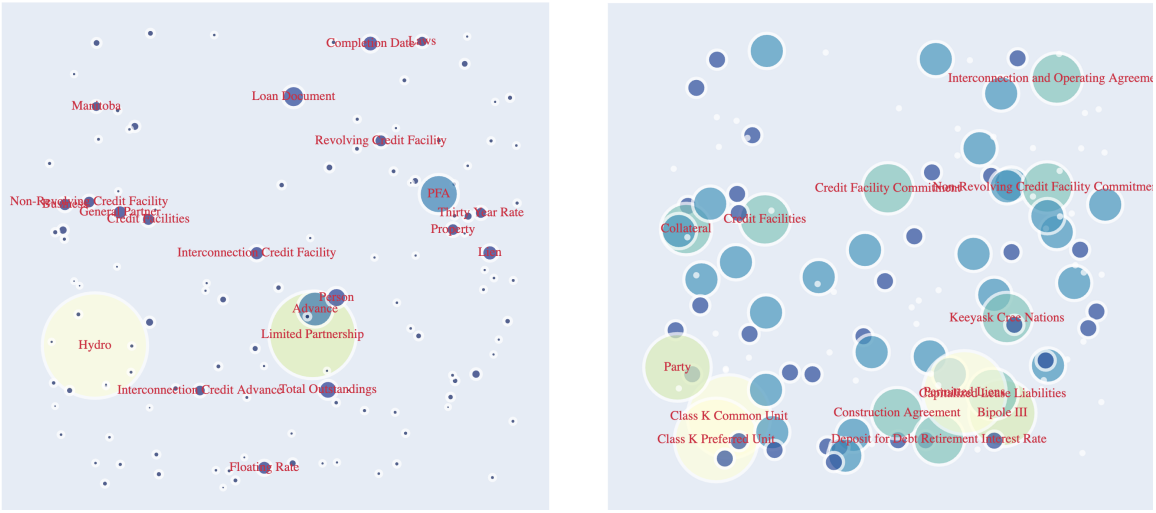

Figure 21: Comparison of WDRW weighting (left) vs. DORW weighting scheme (right) on a short (150 page) Project Finance Agreement.

of definition text bodies compared to section text bodies. The former serves primarily to specify an entity or concept (as elaborated in Chapter 1), such as "Permitted Indebtedness", while the latter serves to specify a relation, such as "Company may take on Permitted Indebtedness". Thus, we hypothesize that the section text body constantly refers to the boilerplate terms as it specifies the relations, since the boilerplate term is generally the head or tail of the relation. For example, a contract between *Company* and *Acquirer* serves to reason out some future states of the world, rights, responsibilities, and remedies between the parties. Thus, the language will often contain language like "to the extent that Acquirer does not...", "if the Company does this...", "if the Acquirer does that then", etc, therein making references to the boilerplate terms more often than any other term. Conversely, in the specification of defined terms, there is no need to be *prescriptive* (and thus make reference to the boilerplate), as such terms need only describe an entity or concept; as such, we posit that this means, like sections, that the most important terms get references most, rather than boilerplate terms.

The precise reasoning for this apparent set disjointedness would make interesting study for future work. For example, we could use an existing Open Information Extraction system (as reviewed in Chapter 2) in order to determine whether the body of the sections is indeed prescriptive, as our intuition suggests. For the rest of this work, we take this as an interesting anomaly, using it to construct an algorithm, but do not further investigate the underlying cause.

CHAPTER 6 : Algorithmic Development

In this chapter, we show results on a preliminary algorithm derived from the visualizations in the previous chapter. With the help of a legal expert, we annotate the top five sections and fifteen defined terms in a subset of our test set. We then use the DORW algorithm for defined terms and the simple cross-reference algorithm for sections as described in Chapter 5. For each of twenty-five sampled documents in the test set (gathered as described in Chapter 4) we use these algorithms to rank the defined terms and sections, respectively, from most important to least important. This ranking corresponds to the size of the nodes in the figures in Chapter 5. We note that the average number of defined terms in our sample was 155, and the average number of sections was 66.

As a baseline ranking for comparison, we rank defined terms and sections by the number of characters they contain. For example, if a defined term d_1 contains one hundred characters, and another d_2 contains fifty, d_1 will be ranked higher than d_1. We show results of our algorithm compared to the baseline in a ROC plot in Figure 22.

On defined terms, the DORW ranking achieves a AUC of 0.62 while the baseline achieves an AUC of 0.66. DORW outperforms for lower (bottom left) areas of the plot, which may be of more practical use for downstream tasks. We also noticed that DORW tended to produce more sensible outputs, while length-weighting was more sensitive to errors in the upstream extraction process. In one example, where the extractor system had erroneously extracted a large defined term body, this was reflected in the baseline extractor. In future work, we will explore combinations of these algorithms in addition to other factors that may improve performance. On

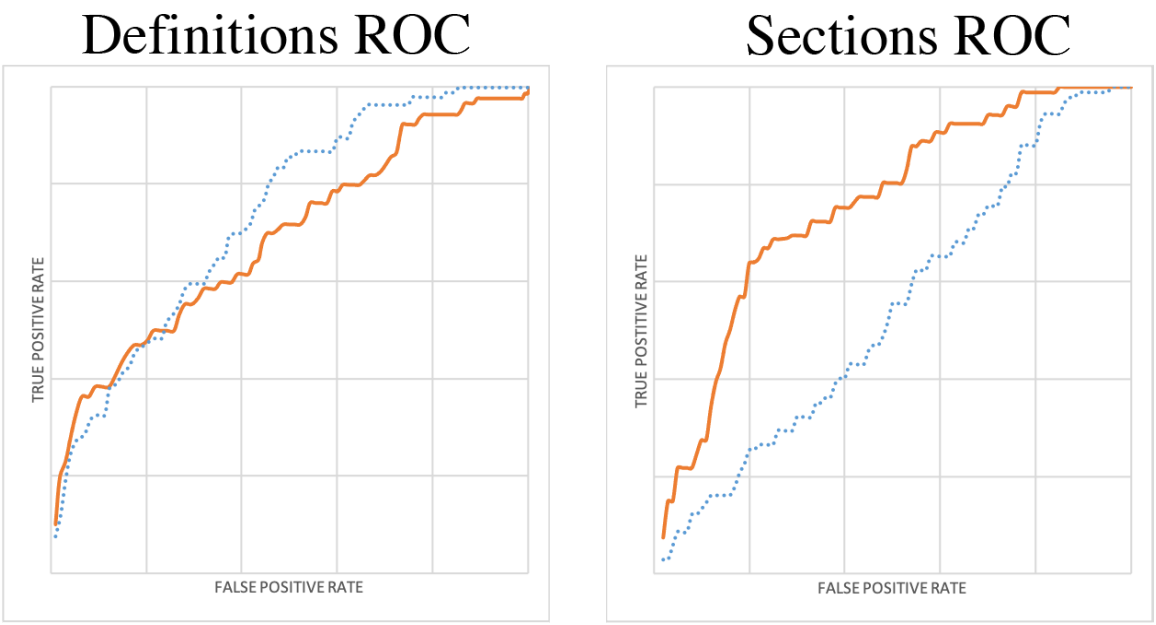

Figure 22: Plot showing ROCs on a domain of [0,1] and range of [0,1]. Dashed blue line is the baseline letter-count algorithm, while orange is the graph-based algorithm. Left shows an ROC plot for defined terms using WDRW weighting. Right shows ROC for sections using simple reference weighting.

sections, the reference-weighting algorithm achieves an AUC of 0.70 while the baseline achieves 0.45.

CHAPTER 7 : Discussion and Future Work

In this work we showed that the syntax used in contracts — the pseudo-formalism called *legalese* — has an underlying graphical structure. While this may be known to some practitioners in the field, this had previously been unexplored in the literature on NLP of contracts as discussed in Chapter 2.

This substructure of contracts, we showed, was not unique to a single agreement, but rather pervasive across a sample of the agreements we analyzed and visualized. Not only did they share a common representation, but the graph presented a new modality for analysis with some common features.

As an academic investigation into language, the conclusion that *contracts exhibit a graph structure* is interesting but not useful by itself. Thus, we then explored the space of algorithms that can operate on this graph. To do this, we took advantage of contemporary graph visualization techniques in Python's NetworkX and Plotly packages. In this process, we showed the following:

1. **There exists and underlying graph structure, and that structure can be made explicit (Chapter 1, 5).** Our first and most important contribution was to show that the graph structure not only exists, but can be extracted and serialized to XML format, analyzed, and results presented to the user. We showed that this was possible in a matter of seconds, not minutes as demonstrated in the prior work Jacob Beckerman (2020), and thus could be of practical benefit for people reviewing legal contracts.

2. **Unconnected nodes in the graph of sections are generally boilerplate (Chapter 5)**. In our visualizations in Chapter 4, we found a common theme

that nodes that are unconnected to the rest of the graph correspond to nodes that a practitioner considers unimportant.

3. **In the graph of defined terms, the weighting scheme matters significantly (Chapter 5)**. We found two different weighting schemes that produced a nearly disjoint set of large nodes in our visualization.

4. **WDRW weighting finds boilerplate defined terms (Chapter 5).** We found that weighting references by the whole document meant that more referenced nodes represented boilerplate. This is not a surprising result to practitioners, as those terms that are used most often are generally the ones that get copy-pasted from deal to deal and are placeholders for different parties to the agreement.

5. **DORW weighting finds defined terms that generally include the more important ones (Chapter 5)**. We found that by limiting the scope of references to those references that are included in the definition body of other defined terms, we can identify important defined terms in our visualizations. This was a surprising result; our ex-post explanation was that defined terms tend to be self-contained and modular, and hence reflect the general behavior of sections, wherein the most important are referenced most often.

6. **We showed that an algorithm to extract key defined terms and sections is possible by establishing a baseline (Chapter 6).** Using the intuitions gained in Chapter 5, we developed a preliminary algorithm which showed good results on a small set of ground-truth-annotated documents.

As a preliminary work in this new formulation of contract analysis, this work proposed more questions than it gave answers. Most importantly, we showed that this is a viable

area for future work, and that algorithmic improvements could yield real-world time savings for practitioners. We propose a few areas for future work:

1. Improve the performance and the accuracy of the core parsing algorithm, yielding better results for all downstream tasks. While the accuracy of our system was sufficient to demonstrate utility, it would benefit from further refinement before being integrated into downstream systems.

2. Further explore the graph structure of contracts on a larger scale. We provided visualizations of a small set of contracts, but no inter-contract visualizations. It would be interesting to visualize the structure of a set of contracts, for example a due diligence package. In order to do this, the core parsing algorithm would need to link across different documents.

3. Develop better summarization and other importance-weighted algorithms. We provided an initial baseline algorithm. One of the issues with this algorithm is that items that are important but are only referenced from one other important definition are not included. A PageRank or similar algorithm that spreads importance from important nodes could potentially solve this problem.

4. As financial research application area, examine the graph structure as a proxy for complexity in modelling. Financial research such as Almeida and Philippon (2007) and Ganglmair and Wardlaw (2015) have done a wide body of research on how complexity of credit and other agreements effects default, performance, or other events. As a metric for complexity, these documents often use semantic measures or word count. Perhaps another good metric for complexity of the deal structure is the number of defined terms, sections, and references; indeed, we did find that the graphs of more complex documents are visually much more

complex. The benefit of this approach is it relies on no semantics, and thus can apply across languages or across deal types (for example, the complexity of a credit agreement may be compared against the complexity of a merger agreement, which is difficult with many semantic-based approaches). We highlight this as just one potential application area for financial research, but imagine there are more research topics beyond our knowledge in legal research, public policy, and clean technology.

BIBLIOGRAPHY

A. V. Aho and M. J. Corasick. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340, June 1975. ISSN 0001-0782. doi: 10.1145/360825.360855. URL https://doi.org/10.1145/360825.360855.

H. Almeida and T. Philippon. The risk-adjusted cost of financial distress. *The Journal of Finance*, 62(6):2557–2586, 2007.

N. Bansal, A. Sharma, and R. K. Singh. A review on the application of deep learning in legal domain. In *IFIP Advances in Information and Communication Technology*, pages 374–381. Springer International Publishing, 2019. doi: 10.1007/978-3-030-19823-7_31. URL https://doi.org/10.1007/978-3-030-19823-7_31.

S. R. C.J. van Rijsbergen and M. Porter. New models in probabilistic information retrieval. 5587, 1980.

J. Donnelly and A. Roegiest. The utility of context when extracting entities from legal documents. In *Proceedings of the 29th ACM International Conference on Information amp; Knowledge Management*, CIKM '20, page 2397–2404, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368599. doi: 10.1145/3340531.3412746. URL https://doi.org/10.1145/3340531.3412746.

B. Ganglmair and M. Wardlaw. Measuring contract completeness: A text based analysis of loan agreements. 2015.

S. H. J. X. Jacob Beckerman, Josh Doman. *University of Pennsylvania Computer and Information Science Senior Design Project*, 1(1), 2020.

M. J. Knuth, Donald E. and V. R. Pratt. Efficient string matching: An aid to bibliographic search. *SIAM Journal on Computing*, 6(2):323–350, June 1974.

C. Niklaus, M. Cetto, A. Freitas, and S. Handschuh. A survey on open information extraction. *CoRR*, abs/1806.05599, 2018. URL http://arxiv.org/abs/1806.05599.

M. R. Roberts and M. Schwert. Interest rates and the design of financial contracts. 2020. URL http://dx.doi.org/10.2139/ssrn.3508816.

J. Stewart, G. Ciany, and J. Carbonell. Genre identification and goal-focused summarization. pages 889–892, 01 2007. doi: 10.1145/1321440.1321568.

W. Wei, A. Roegiest, and M. Mikhail. From bubbles to lists: Designing clustering for due diligence. In *Proceedings of the 2019 Conference on Human In-*

formation Interaction and Retrieval, CHIIR '19, page 277–281, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450360258. doi: 10.1145/3295750.3298951. URL https://doi.org/10.1145/3295750.3298951.