Semantic Information Extraction from Natural Language using a Learning and Rule-based Approach

by

Varun Singh

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2023 by the
Graduate Supervisory Committee:

Srividya Bansal, Chair
Ajay Bansal
Alexandra Mehlhase

ARIZONA STATE UNIVERSITY

December 2023

ABSTRACT

Open Information Extraction (OIE) is a subset of Natural Language Processing (NLP) that constitutes the processing of natural language into structured and machine-readable data. This thesis uses data in Resource Description Framework (RDF) triple format that comprises of a subject, predicate, and object. The extraction of RDF triples from natural language is an essential step towards importing data into web ontologies as part of the linked open data cloud on the Semantic web. There have been a number of related techniques for extraction of triples from plain natural language text including but not limited to ClausIE, OLLIE, Reverb, and DeepEx. This proposed study aims to reduce the dependency on conventional machine learning models since they require training datasets, and the models are not easily customizable or explainable. By leveraging a context-free grammar (CFG) based model, this thesis aims to address some of these issues while minimizing the trade-offs on performance and accuracy. Furthermore, a deep-dive is conducted to analyze the strengths and limitations of the proposed approach.

DEDICATION

Dedicated to my family, and my best friend and companion, Spidey.

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor Dr. Srividya Bansal for her continued guidance and support throughout my time at ASU. I would like to thank my committee members Dr. Ajay Bansal and Dr. Alexandra Mehlhase for their constructive feedback and valuable insights. I would like to extend my gratitude to Smruthi, for her endless encouragement, love, and support.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

With the growth of the semantic web and knowledge graphs, there is a dire need for data to be processed and formatted into triples to allow integration into semantic databases and linked open data clouds. Significant amounts of data are in the form of documents in natural language that can be leveraged to add to these semantic knowledge bases. There is a need for novel approaches for effective and accurate extraction of semantic triples from natural language.

1.1 Motivation

There has been a lot of research in the area of Open Information Extraction (OIE) since the late 2000s (1). The challenge has been to convert unstructured information written using natural language into a structured representation. Different approaches have been taken to solve this problem including learning-based approaches, rule-based approaches, and clause-based approaches (2). With the growth of data on the web and knowledge graphs, there is a greater need for the conversion of text into structured formats. Semantic web technologies such as the Resource Description Framework (RDF) data model use a triple format to represent data (3). The triple comprises of a subject, predicate, and object describing a data resource. The authors of Mold framework (4) focused on the extraction of entities (subject and object) from given unstructured text as well as producing summarization paragraphs from given structured triples. Current approaches rely heavily on machine-learning models and/or require humans in the loop to validate the extracted patterns or produce training data. This often required manually tagging instances of seed data. Several approaches focus on a specific domain and cannot be used for different kinds of information and terms. Handling heterogeneous datasets is a huge challenge. This thesis focuses on the extraction of relations or properties for RDF triple generation from unstructured text.

1.2 Problem Definition

This thesis aims to create a novel approach to the task of open information extraction. More specifically, the problem is to come up with an effective approach to convert given unstructured natural language text into RDF triples that can be integrated into the available open knowledge graphs on the web. We aim to address some of the issues with the current state of conventional machine learning (ML) methods.

Lack of training data - Traditional ML models require large annotated datasets for training the models. This is a challenge when such datasets are not readily available, especially for domain-specific data. The latest advancements in Large Language Models (5) have produced a lot of trained models for general-purpose text. However, this remains a challenge for specific domains.

Customization - Most ML-based information extraction systems have been trained on a wide variety of data to perform well in most situations. However, for domain-specific tasks, or for tasks where there might be some deviations from the standard grammar (of the language), it can be very expensive, resource-intensive, and may not even be feasible to tweak ML models to account for such changes.

Support for additional languages - While some languages have many related tools and datasets, that may not be true for others. This makes building information extraction systems for them even more challenging (6).

Explainability - Deep learning and other complex ML models are effectively a black box. While they can generate high-performance metrics, it is extremely difficult to understand the logic and reasoning behind any given output. This leads to issues with trust and accountability (7).

1.3 Hypothesis

The proposed model aims to use a hybrid approach that includes machine learning models and rules in the form of context-free grammar to convert natural language data into semantic RDF triples. Our hypothesis is that this combined approach would address some of the challenges with respect to explainability, customization of data for specific domains, and support for additional languages. Rules will help with necessary customizations for specific domains or languages. The use of existing machine learning models helps with the initial processing of the text and its tokenization. The triples produced can be further processed and stored in the semantic web using ontologies. The main problems this approach aims to address are as follows:
Lack of training data - Since the model will be built for a particular language with known rules of usage, there is no need for any prior data to be maintained for training purposes.
Customization - Rules in the model can be modified to suit the needs of a specific domain or to account for certain colloquial styles of language that may include incorrect use of grammar. With this model, customization can be achieved at a relatively low cost, without the need for extensive training or overfitting to a particular dataset.
Support for additional languages - In theory, this approach can be extended to

parse text from other languages as well. One would require a deep understanding of the grammar and usage of the language to create the set of rules for the parser. However, this approach is limited by the availability of a reliable part-of-speech (POS) tagger for the language in question.

Explainability - A significant problem with machine learning models is that their results are not explainable. This makes them inherently unreliable even though they may have excellent performance results. Using a Logic Programming language such as a Prolog program with Definite Clause Grammar (DCG) rules, it is possible to generate the exact sequence of rules that passed in order to generate a given triple.

The remainder of this thesis is organized as follows. Chapter 2 covers the background information relevant to the study. It includes the basics of the Semantic Web, Ontologies, and Natural Language Processing. It also includes related work in the area of Open Information Extraction. Chapter 3 describes the proposed approach, high-level design, and rules for conversion. Chapter 4 describes the datasets used in the study and processing of data. Chapter 5 presents the experiments and evaluation of the proposed approach. Chapter 6 presents Future work followed by conclusions in Chapter 7. All the references are listed after this chapter.

Chapter 2

RELATED LITERATURE

2.1 Background

This section presents the necessary background on the Semantic Web (RDF data model, Ontology, technology stack) and Natural Language Processing tools used in this thesis.

2.1.1 The Semantic Web

The World Wide Web Consortium (W3C) is building a “web of linked data” as an extension to the current World Wide Web, with the end goal of making all web resources machine-readable (8). This involves machine-processible meta-data for all web pages, web services, and data on the web. This includes the need for open information extraction to produce structured data. It also includes the use of ontologies to define domain-specific vocabulary and connect data to corresponding classes and properties in the ontology. Here are some of the important background concepts.

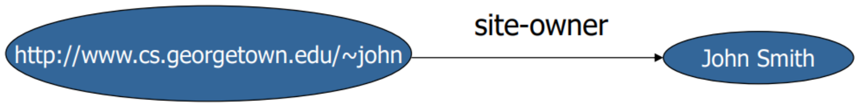


Figure 2.1: Graph Notation of a Triple

- **Triple**

  A triple is an entity of semantic information in the form of a subject, predicate, and an object. This format is specified by the Resource Description Framework (RDF). A triple is also known as a semantic triple, or an RDF triple (3). RDF provides a graph data model to describe web resources and provide links between resources thereby forming a knowledge graph of information. This knowledge graph consists of schema information in the form of ontologies and instance information in the form of RDF triples all form one large knowledge graph that can be queried. The open data on the web in this format comprises the linked open data cloud (9) with billions of triples from various domains along with their ontologies.

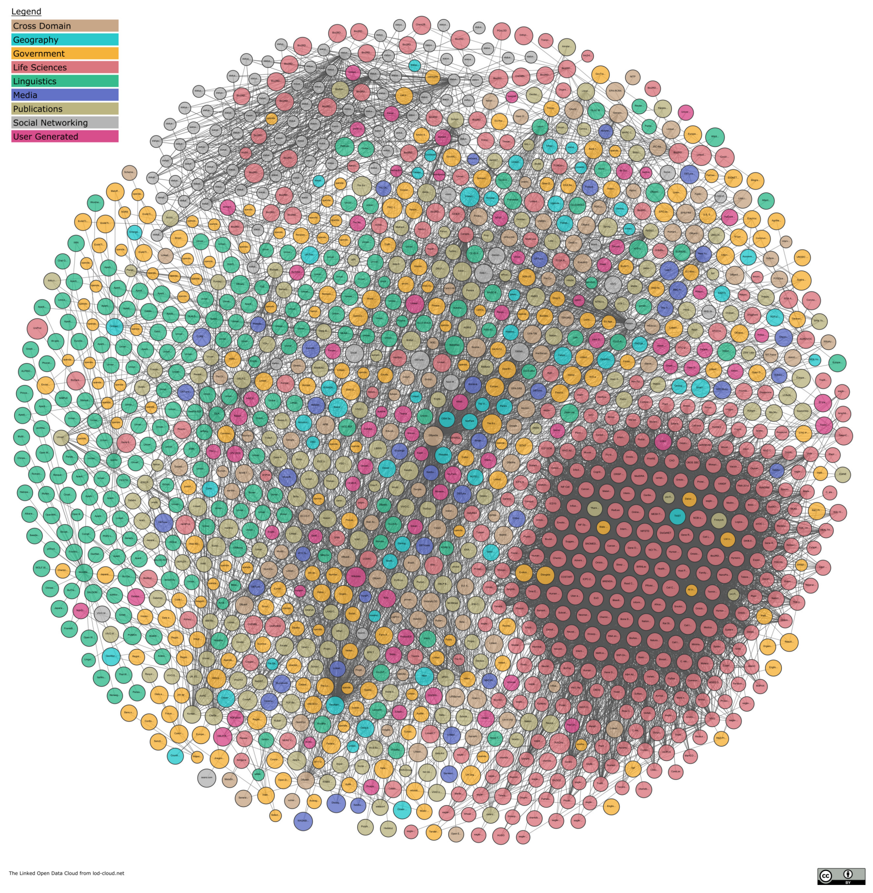


Figure 2.2: Linked Open Data Cloud, Taken from Lod-cloud.Net

- **Ontology**

  The OWL Web Ontology Language is a markup language that is used for publishing and sharing ontologies. OWL is built upon RDF and an ontology created in OWL is a RDF graph. Individuals with common characteristics can be grouped to form a class. OWL provides different types of class descriptions that can be used to describe an OWL class. OWL also provides two types of properties: object properties and data properties. Object properties are used to link individuals to other individuals while data properties are used to link individuals to data values. OWL enables users to define concepts in a way that allows them to be mixed and matched with other concepts for various uses and applications. Protégé is an open-source ontology editor and framework for building intelligent systems (10). It allows users to create ontologies in W3C's Web Ontology Language. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional constructs along with formal semantics (11).

2.1.2 Natural Language Processing

Natural Language Processing (NLP) is a subset of machine learning that is concerned with enabling computers to interact directly with natural language rather than machine-readable data. This thesis is focused on extracting entities and relations from unstructured natural language to generate semantic RDF triples. Several existing NLP libraries handle the processing of human language through a pipeline of processes that include analyzing and understanding the text. The steps in the pipeline include the acquisition of data, cleaning of text, pre-processing, feature engineering, evaluation, model building, and deployment.

The pre-processing steps typically include tokenization, lower casing, removal of

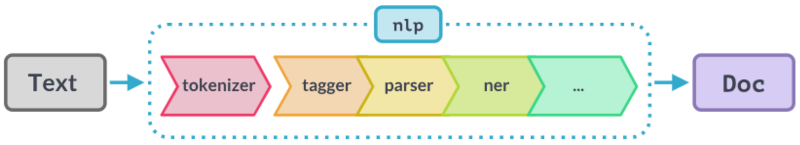


Figure 2.3: NLP Pipeline Example

stop words, stemming or lemmatization, removal of punctuation, and part-of-speech tagging followed by named-entity recognition. Within the scope of this thesis, we will primarily look at tokenization and part-of-speech (POS) tagging.

Tokenization is the process of splitting a raw sentence down to its individual words or tokens. The tokens can be further lemmatized to get the base form of their word, as in a dictionary, but we will not be doing that since we want the tokens from the sentence as is, in line with the task of open information extraction.

Part-of-Speech (POS) tagging is the process of assigning tags to each token that denotes its function with respect to the grammar of the English language. Here, we will go over the list of POS tags (12).

- **ADJ** - Adjective

  Adjectives are words that typically modify nouns and specify their properties or attributes.

  For example, The *oldest French* bridge

- **ADP** - Adposition

  Adposition is a cover term for prepositions and postpositions. Adpositions belong to a closed set of items that occur before (preposition) or after (post-position) a complement composed of a noun phrase, noun, pronoun, or clause that functions as a noun phrase, and that forms a single structure with the complement to express its grammatical and semantic relation to another unit

within a clause.
For example, *in*, *to*

- **ADV** - Adverb
  Adverbs are words that typically modify verbs for such categories as time, place, direction, or manner. They may also modify adjectives and other adverbs.
  For example, *arguably* wrong.

- **AUX** - Auxiliary
  An auxiliary is a function word that accompanies the lexical verb of a verb phrase and expresses grammatical distinctions not carried by the lexical verb, such as person, number, tense, mood, aspect, voice, or evidentiality.
  For example, He *is* a teacher.

- **CCONJ** - Coordinating Conjunction
  A coordinating conjunction is a word that links words or larger constituents without syntactically subordinating one to the other and expresses a semantic relationship between them.
  For example, *and*, *or*, *but*

- **DET** - Determiner
  Determiners are words that modify nouns or noun phrases and express the reference of the noun phrase in context.
  For example, *a*, *an*, *the*, *this*

- **INTJ** - Interjection
  An interjection is a word that is used most often as an exclamation or part of an exclamation. It typically expresses an emotional reaction, is not syntactically related to other accompanying expressions, and may include a combination of

sounds not otherwise found in the language.

For example, *ouch*, *bravo*, *hello*

- **NOUN** - Noun

  Nouns are a part of speech typically denoting a person, place, thing, animal, or idea.

  For example, *girl*, *tree*, *decision*

- **NUM** - Numeral

  A numeral is a word, functioning most typically as a determiner, adjective, or pronoun, that expresses a number and a relation to the number, such as quantity, sequence, frequency, or fraction.

  For example, *100*, *two*

- **PART** - Particle

  Particles are function words that must be associated with another word or phrase to impart meaning and that do not satisfy definitions of other universal parts of speech.

  For example, *'s*, *not*

- **PRON** - Pronoun

  Pronouns are words that substitute for nouns or noun phrases, whose meaning is recoverable from the linguistic or extralinguistic context.

  For example, *she*, *we*, *they*

- **PROPN** - Proper Noun

  A proper noun is a noun (or nominal content word) that is the name (or part of the name) of a specific individual, place, or object.

  For example, *Mary*, *London*, *UN*

- **PUNCT** - Punctation

  Punctuation marks are non-alphabetical characters and character groups used in many languages to delimit linguistic units in the printed text.

  For example,

  Period: . Comma: ,

- **SCONJ** - Subordinating Conjunction

  A subordinating conjunction is a conjunction that links constructions by making one of them a constituent of the other. The subordinating conjunction typically marks the incorporated constituent which has the status of a (subordinate) clause.

  For example, *if*, *while*

- **SYM** - Symbol

  A symbol is a word-like entity that differs from ordinary words by form, function, or both.

  For example, *$*, *%*

- **VERB** - Verb

  A verb is a member of the syntactic class of words that typically signal events and actions, can constitute a minimal predicate in a clause, and govern the number and types of other constituents that may occur in the clause. Verbs are often associated with grammatical categories like tense, mood, aspect, and voice, which can either be expressed inflectionally or using auxiliary verbs or particles.

  For example, *eat*, *ate*, *eating*

- **X** - Other

The tag X is used for words that for some reason cannot be assigned a real part-of-speech category. It should be used very restrictively. These are usually words that do not belong to the English language and also cannot be tagged as proper nouns.
For example, And then he just *xfgh*

2.1.3 Key Technologies

In this section, we will go over some of the programming languages and external packages that we have used to implement our context-free grammar (CFG) model.

- **SWI Prolog (version 8.4.2)**

  Prolog is a high-level logic programming language associated with artificial intelligence and computational linguistics. As a logic-based programming language, Prolog provides in-built support for reasoning, and backtracking, allowing it to handle complex tasks. Within the scope of this thesis, Prolog plays a significant role in the development of our model with its robust support for context-free grammar due to its Definite Clause Grammar (DCG) syntax (13).

- **Python (version 3.9.9)**

  Python is a high-level programming language that is known for its expansive ecosystem of built-in libraries and wide range of applications. We have chosen this language because it supports object-oriented development of our application, and also supports interactions with the external tools related to natural language processing and Prolog that is required for this thesis(14).

- **pyswip (version 0.2.10)**

  pyswip operates as a bridge between the main python controller and the Prolog file containing the rules for context-free grammar. It allows us to directly make prolog queries from a Python environment which is integral for this task (15).

- **spacy (version 3.2.0)**

  spacy is an industry-strength library that can be used to perform a wide range of tasks in the domain of natural language processing. Due to its high levels of performance, ease of use, and ease of deployment within an end-to-end application, we have decided to use it for performing the necessary NLP operations. Within the scope of our project, we will leverage its capabilities for tokenization and part-of-speech (POS) tagging, which will be part of the data pre-processing phase, serving to create the inputs for our CFG model.

2.2 Related Work in Open Information Extraction

2.2.1 Domain Based OIE Systems

Using a domain-specific ontology, the authors in (16) have shown accurate extraction of named entities and information relevant to the same. In the example, they have used an ontology for Hotels, that deals with objects of a Hotel class and its attributes such as rooms and other on-site amenities. Using a dependency parse graph, they have used a breadth-first search graph traversal algorithm to extract the three parts to form a triple. However, the required ontology can vary based on their identification of the domain or main theme of the document. of the document. We need an approach for triple extraction that is not dependent on a given ontology or domain.

There are other works that do not use a domain-specific ontology, as shown by the Mold framework (4). They provide a domain-free framework that could be applied to natural text and in return generate linked data (RDF triples). It certainly helps to have domain experts provide further insight into tailoring the base ontology to their domain. Furthermore, this framework can be leveraged to provide summarizations of the input document(s) or corpus of unstructured text.

Abedini et al. have shown that natural language processing can be used for named entity recognition to find ambiguous entities in natural text. This method relies on the use of an ontology for disambiguation, which improves the accuracy of the extracted entities but also creates a limitation in cases where such entities do not already exist in the knowledge base (17).

2.2.2 Rule Based OIE Systems

The authors in (18) have leveraged linguistic patterns and an onomasiological approach to defining semantic relations. This helps discover the occurrences of the relation in natural language and also if it is consistent with its initial definition (19).

It is interesting to note that the authors in (20) reject the use of lexical patterns and instead rely on a syntactic parser to extract semantic relations, followed by support vector machines to classify them. The results obtained are comparable to the aforementioned approaches.

ClausIE creates a syntactic tree structure that represents a sentence using the Stanford dependency parser. Potential types of clauses are identified that match with their seven pre-defined clauses - SV, SVA, SVC, SVO, SVOO, SVOA, SVOC. Here, S is subject, V is verb or predicate, O is direct object, C is complement, and A is adverbial. These clause types are matched to natural language by using the dependency relations between tokens. In the end, the set of triples derived from that sentence is generated based on the different combinations of subject and object constituents (21).

ReVerb uses POS tags, Noun Phrase (NP) chunks, and syntactic constraints to identify relations in a sentence around verbs. Then, it extracts potential arguments for the relation by looking for noun phrases closest to the identified verb, on the left and right sides of the verb respectively. Furthermore, they use a logistic regression classifier trained on a dataset of manually annotated sentences and triple extractions, to filter out incorrect triples. Our approach follows a similar pattern, by using context-free grammar (CFG) to parse the sentence into triples. We aim to achieve correct triple extractions based on the CFG, removing the need for a classifier to filter out incorrect results (22).

PROPS focuses on semantic representation over syntactic detail by outlining five principles (23): Masking non-core syntactic detail: It simplifies the representation by removing auxiliary words and grouping atomic units. Representing propositions uniformly: It aims to cover a wide range of propositions, not just verb-centric ones. Canonicalizing and differentiating syntactic constructions: The goal is to unify semantically equivalent propositions and distinguish between syntactically similar but semantically different constructions. Marking proposition boundaries: This highlights the span of standalone propositions and their elements. Propagating Relations: All inferable relations through parse tree traversal are explicitly marked.

2.2.3 Learning Based OIE Systems

Yotedje (24) has demonstrated a new system called Generic Information Extraction using Triple Store databases (GIET) where they extract information from natural language in response to a given query in natural language. Using entity extraction, and dependency parse tags, they store the entity type, parse, tag, and word position in a triple-store database. This is then queried using SPARQL to extract meaningful information based on the query posted to the GIET system.

Another learning based system called Open Language Learning for Information

Extraction (OLLIE) (25) builds upon the foundations of ReVerb, previously mentioned in the rule-based systems. Using ReVerb extractions as training data, OLLIE learns the mapping between ReVerb's template patterns and their corresponding triple extraction for the dependency parse tree of the sentence.

Starting with a set of template patterns, NestIE learns to recognize their variations. Using this knowledge, it can analyze the dependency tree structure of a sentence and map it to valid propositions. Moreover, from each extracted proposition, it can expand further to find nested propositions, if they exist. As defined by the authors, a proposition is synonymous with a triple i.e., a relation of the form (Object1, relation, Object2) (26).

The methodology proposed by the authors of DeepEx (27) uses two primary stages: generation and ranking. In the generation phase, the pre-trained language models are used to produce possible triples from the input. In the ranking stage, these triples are processed to ensure relevance and relational integrity using a contrastive model that leverages a BERT encoder. The T-REx dataset (28) is used to train the model, which provides large-scale alignments between Wikipedia abstracts and Wikipedia triples. The final output is a set of the most suitable triples extracted from the input text.

The CASREL framework focuses on relational triple extraction from sentences, identifying (subject, relation, object) sets directly. CASREL maximizes the data likelihood of the training set by checking all three components of the triple, instead of treating entities and relations separately. This methodology establishes a novel tagging scheme that facilitates direct optimization, handles overlapping triples, and views each relation as a function mapping the subjects to objects. The process of extraction uses a subject tagger to identify subjects and, for each subject, relation-specific object taggers locate the compatible relations and objects. Both taggers are built atop the BERT Transformer, a deep bidirectional text encoder. The framework uses a two-step Cascade Decoder which first spots probable subjects and detects related objects within the context of these subjects. The entire model is trained to maximize a data log-likelihood objective using the Adam optimization method (29).

The authors present a method to extract triples by first converting natural language into relational graphs using a bi-directional Pretrained Language Model (PLM). Each word is encoded into a vector, and these are structured into a graph to capture relations. From this graph, an autoregressive PLM is used to regenerate text, following which the relational triples are extracted using the aforementioned CASREL approach (30).

EmRel is a system designed to derive valid triples from text by aligning entity and relation structures. First, it uses tools such as BERT to parse entities and embed relations. It merges these representations using a multi-head attention mechanism, ensuring they are analogous. Lastly, the system scores the validity of each potential triple through the Tucker decomposition and refines these scores using a cross-entropy loss function (31).

We have extensively looked into defining patterns based on grammar including simple and compound verbs. We can adopt powerful tools used for natural language processing that include part-of-speech tagging and dependency resolution, to allow us to extract triples and examine relations based on the generated knowledge graph. Additionally, one or more ontologies can be used for disambiguation purposes to validate the extracted relations. Some of the related works extensively use machine learning, which we plan to minimize with our proposed context-free grammar (CFG) model. In the next chapter, we will go over our approach and methods in detail.

Chapter 3

PROPOSED APPROACH AND HIGH-LEVEL DESIGN

The approaches taken so far for information extraction and triple generation include learning-based approaches that need training models and data or domain-specific approaches that only work for a specific domain or need an ontology that describes the domain. Our proposed approach uses a combination of learning-based and rule-based approaches. It leverages the existing highly developed and leading machine learning approaches for Natural language processing to parse a given sentence by tokenizing it and adding parts of speech tags to the tokens. Then we use a rule-based approach to apply rules on the the tokens in order to identify the potential subject, predicate, and objects for the RDF triple. This involves coming up with detailed rules taking into consideration all aspects of constructing an English sentence and its grammar. The following sections present the design of our study, the natural language processing pipeline, definite clause grammar rules, and the high-level system design.

3.1 Overall Approach

Context-free grammar (CFG), from formal language theory, provides formal grammar with rules that can be applied to nonterminal symbols irrespective of their context. CFG comprises of tokens or terminal symbols and non-terminal symbols. It also has production rules with nonterminals on the left-hand side and a list of terminal or nonterminal symbols on the right-hand side of the rule. Definite clause grammar (DCG) is a way to express CFG rules in the logic programming language Prolog. The overall approach uses DCG to provide rules for converting extracted tokens from natural language into RDF triples. Firstly, we have the natural language processing pipeline using an industrial-strength open-source library in Python called spacy (32). Bidirectional Encoder Representations from Transformers (BERT) is a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based on their connection. It is an open-source machine learning framework commonly used for NLP tasks to understand the meaning of ambiguous language in text corpora. The NLP pipeline consists of a tokenizer followed by a part-of-speech (POS) tagger. It is common to also include a lemmatizer following the tokenization process, we have chosen to ignore that as we are looking to extract tokens as they occur in natural language. Internally, spacy uses RoBERTa (33), an optimized version of the BERT model that we will use to attach POS tags to the tokens in a sentence. Next, the sequence of tokens with POS tags will be passed to a Prolog program. This program includes a list of definite clause grammar (DCG) rules that is an implementation of the context-free grammar (CFG) model. This will generate one or more sets of triples after being parsed by the CFG model. These are then matched against the valid triples to evaluate the performance of the CFG model.

A natural language sentence can be defined as a list of tokens. Given a list of tokens T, the extraction task is to form a meaningful triple <S,P,O> where <S,P,O> each is a list of tokens such that each token in S,P,O belongs to T, and the order of these tokens may deviate from that in T, The triple <S,P,O> captures some meaningful information of the form of <object1, relation, object2>. S and O are generally entities, but that is not necessary since this is an open information extraction task.

3.2 NLP Pipeline

As part of the spacy NLP library, we have used the built-in model denoted as 'en_core_web_lg'. This model has been trained for a wide variety of tasks including tokenization, POS tagging, lemmatization, dependency parsing, and named entity recognition. This model has been trained on some of the more popular datasets such as Ontonotes 5.0 (34), and GloVe Common Crawl (35). Ontonotes 5.0 is a large-scale, multi-genre, multi-lingual corpus that has been manually annotated with syntactic and semantic information. It has data on news, weblogs, telephone conversations, talk shows, broadcasts, and Usenet group messages in English, Chinese, and Arabic. Glove is global vectors for word representation that is created based on the co-occurrence of words obtained from statistics of a huge corpus of Wikipedia data. The NLP pipeline used consists of a tokenizer and a part-of-speech (POS) tagger. The tokenizer takes the raw sentence as input and breaks it into a list of tokens. These tags denote the function of the word in the sentence. This forms the initial input that can be parsed by the context-free grammar (CFG) model.

Example sentence: A graduate of the University of Arizona, she is a lecturer in writing at Princeton University and an author under the name of Kathryn Watterson Burkhart.

Example output: [['A', 'DET'], ['graduate', 'NOUN'], ['of', 'ADP'], ['the', 'DET'], ['University', 'PROPN'], ['of', 'ADP'], ['Arizona', 'PROPN'], [',', 'PUNCT'], ['she', 'PRON'], ['is', 'AUX'], ['a', 'DET'], ['lecturer', 'NOUN'], ['in', 'ADP'], ['writing', 'NOUN'], ['at', 'ADP'], ['Princeton', 'PROPN'], ['University', 'PROPN'], ['and', 'CCONJ'], ['an', 'DET'], ['author', 'NOUN'], ['under', 'ADP'], ['the', 'DET'], ['name', 'NOUN'], ['of', 'ADP'], ['Kathryn', 'PROPN'], ['Watterson', 'PROPN'], ['Burkhart', 'PROPN'], ['.', 'PUNCT']]

The above output shows how the example sentence is broken up into tokens and each token has been tagged indicating what part-of-speech it is. An explanation of the individual tags was presented in section 2.1.2.

3.3 Prolog DCG

Prolog, a Logic Programming language, provides an ideal environment for the implementation of the CFG model. The Definite Clause Grammar (DCG) syntax provides a clear and concise representation of the grammar rules and also supports built-in mechanisms for parsing. DCG is a programming construct or expression available in Prolog and similar logic programming languages.

DCG is defined as a rule in Prolog that comprises of a Head and Body as follows:

Head –> Body.

DCGs have the general form

non-terminal –> d1, ..., dN.

where d1 through dfN are either non-terminals or terminal symbols. Drawing parallel with natural language, non-terminals would be similar to nouns and verb phrases, while terminals resemble actual words. This makes Prolog an ideal solution for this task.

3.4 CFG Rules

The coverage and applicability of the DCG rules determine the performance and quality of the CFG model Initially derived from linguistic and syntactic patterns, these rules serve as guidelines for the system to determine potential triples within a sentence. It will become clear that these rules somewhat differ from the natural production rules for the English language, this is because the objective of both grammars is different. While the former only checks if a sentence is grammatically correct, we are looking to extract certain parts of a sentence to create meaningful triples. These rules have been further refined in an iterative process, based on multiple rounds of testing followed by the adjustment of rules to improve performance and accuracy. For the purpose of maintaining similarity between the grammar rules and their Prolog implementation, terminal symbols will be in uppercase, and non-terminal production rules will be in lowercase.

In the following section, we present rules for identifying various parts of a triple, that is, subject, predicate, and object given tokens of a natural language sentence as input.

3.4.1 Subject Rules

In this subsection, we will go over the rules to parse a subject entity.

term0 and term1 denote the atomic blocks that are used to build a subject entity. The terminal symbols of a subject rule are noun (NOUN), pronoun (PRON), proper noun (PROPN), adjective (ADJ), symbol (SYM), and number (NUM).

term0 -> PROPN
term0 -> PROPN '.'
term0 -> NOUN
term0 -> NUM
term0 -> ADJ
term0 -> term0 term0

Table 3.1: Term0 Rules

term1 also includes terminals, but this is mainly done to separate pronouns and currency from term0.

term1 -> PRON
term1 -> SYM NUM
term1 -> term0

Table 3.2: Term1 Rules

The rule sets from term2 to term6 gradually allow more combinations and build complexity of the possible subject entities. This is similar to how mathematical expressions are parsed by compilers in most programming languages - numbers and variables form the atomic blocks or terminal symbols, and the hierarchical structure of rule sets progressively adds more types of mathematical operators.

term2 -> ADJ term2
term2 -> VERB term2
term2 -> term1

Table 3.3: Term2 Rules

term2 accounts for chaining multiple adjectives and also for adjectives that are based on verbs.

term3 -> ADV term2
term3 -> term2

Table 3.4: Term3 Rules

term3 handles adverbs (ADV) that come before any adjectives.

term4 -> DET
term4 -> DET term3
term4 -> NUM term3
term4 -> term3

Table 3.5: Term4 Rules

term4 adds determiners (DET) like ”a”, ”an”, ”the” etc., and numbers that are used to denote the quantity or amount of a subject entity.

term5 -> term4 CCONJ sub
term5 –> term4

Table 3.6: Term5 Rules

term5 is used for handling coordinating conjunctions (CCONJ) like ”and”, ”or” etc. This leads to forming more complex subjects that are a combination of two or more subject entities.

term6 -> term5 ADP sub
term6 –> term5

Table 3.7: Term6 Rules

term6 deals with adpositions to create prepositional structures in subject entities.

sub -> term6 PART sub
sub -> " tokens "
sub -> term6

Table 3.8: Sub Rules

The sub ruleset is at the top of the hierarchy of the subject rules encompassing all the others. Additionally, it also works for possessive particles like "'s", and quotes.

3.4.2 Object Rules

In this section, we will go over the grammar rules for parsing objects.

obj -> ADJ
obj -> ADV VERB

Table 3.9: Simple Object Rules

These are the simple rules that define terminals for an object clause.

obj -> sub
obj -> ADP sub

Table 3.10: Subject Related Object Rules

Since object and subject clauses are functionally similar in terms of syntax, we built the subject rules in a way that can also capture object clauses.

obj -> sub CCONJ tokens
obj -> sub SCONJ tokens
obj -> sub ADP tokens
obj -> sub DET tokens
obj -> sub VERB
obj -> sub VERB tokens

Table 3.11: Complex Object Rules

However, a subject clause may contain more information that does not necessarily need to be part of the object. Using the rule "tokens", we can ignore any number of tokens after the object clause, using certain types of tokens as delimiters - coordinating (CCONJ) and subordinating conjunctions (SCONJ), determiners (DET), verbs (VERB), and adpositions (ADP).

3.4.3 Predicate Rules

Here, we will examine the rules for identifying predicate clauses. We have two groups as follows

pred -> VERB
pred -> ADV VERB
pred -> VERB ADP
pred -> AUX

Table 3.12: Simple Predicate Clauses

These are the most basic predicate clauses. Using verbs, adverbs, and auxiliaries, we can create simple predicates.

pred -> VERB pred
pred -> AUX pred
pred -> VERB PART pred
pred -> VERB NOUN PART pred
pred -> AUX PART pred

Table 3.13: Complex Predicate Clauses

These rules are used for more complex predicate clauses that are built by chaining verbs, or modifying them with other types of tokens.

3.4.4 Sentence Rules

Here, we will look at the rulesets used to parse a complete sentence. The 'sent' rule is at the top of the hierarchy and calls other rules previously defined in this chapter.

vp -> pred obj
vp -> pred obj .
vp -> pred obj PUNCT tokens
vp -> tokens PUNCT vp

Table 3.14: Verb Phrases

First, we have the 'vp' ruleset, which is an additional structure we have created to parse verb phrases in the context of the 'sent' rules. These rules identify a predicate P and object O from the verb phrase, which are combined with a subject S from a 'sub' rule to form a triple.

sent -> sub vp
sent -> sub PUNCT vp

Table 3.15: Basic Sentences

These are simple sentences with a subject followed by a verb phrase that consists of a predicate and an object phrase.

sent -> sub PUNCT sent
sent -> obj PUNCT sent
sent -> tokens PUNCT sent
sent -> " sent "

Table 3.16: Nested Sentences

These are nested sentences, or more complicated sentences containing multiple sub-sentences due to the use of punctuations like commas and semicolons. In these cases, the parts of a triple may come from different sub-sentences. Also, the use of tokens rule helps to remove unnecessary tokens at the start of sentences like "However, ...".

sent -> sub pred tokens PUNCT tokens
sent -> sub pred SCONJ tokens .
sent -> sub pred NOUN SCONJ tokens .
sent -> sub pred tokens .

Table 3.17: Sentences with Complex Object Clauses

This is the most generic ruleset for sentences that do not match any other patterns. Based on some predefined delimiters like conjugations and punctuations, we can identify a set of tokens that could potentially be the object phrase, and form a triple using it.

3.5 High-Level System Design

For this study, we have used Python3 for running the main controller, and SWI Prolog for running the context-free grammar (CFG) model. We have used the package pyswip as a bridge between the controller and SWI Prolog in order to directly make Prolog queries to the CFG model. Figure 3.1 shows the high-level system diagram and flow of activities. Each raw sentence in natural language is first passed to the spacy module of the NLP pipeline to tokenize and tag parts of speech of the sentence. This formatted input to passed to the CFG Prolog module that applies the DCG rules to identify subject, predicate, and object in order to generate possible triples. The generated triples are the final output of the system.

Raw sentence	A graduate of the University of Arizona, she is a lecturer in writing at Princeton University and an author under the name of Kathryn Watterson Burkhart.
Output of tokenizer + POS tagger	[['A', 'DET'], ['graduate', 'NOUN'], ['of', 'ADP'], ... ,['Kathryn', 'PROPN'], ['Watterson', 'PROPN'], ['Burkhart', 'PROPN'], ['.', 'PUNCT']]
SWI Prolog query	sent(X, [['A', 'DET'], ['graduate', 'NOUN'], ['of', 'ADP'], ... ,['Kathryn', 'PROPN'], ['Watterson', 'PROPN'], ['Burkhart', 'PROPN'], ['.', 'PUNCT']], []).
Output of CFG model	('she', 'is', 'A graduate') ('she', 'is', 'a lecturer')

Table 3.18: Data at Each Stage of the Application

The table 3.18 shows an example natural language sentence and how it is processed through the pipeline in our proposed system. The raw sentence is first processed using spacy library and tokens are produced. Output from the tokenized is shown. These tokens are then given part-of-speech tags. Next, the tokens with tags are passed to the SQI Prolog query that runs them through the DCG to identify the subject, predicate, and object. The final output from the CFG model is the possible triples generated as shown in the table.

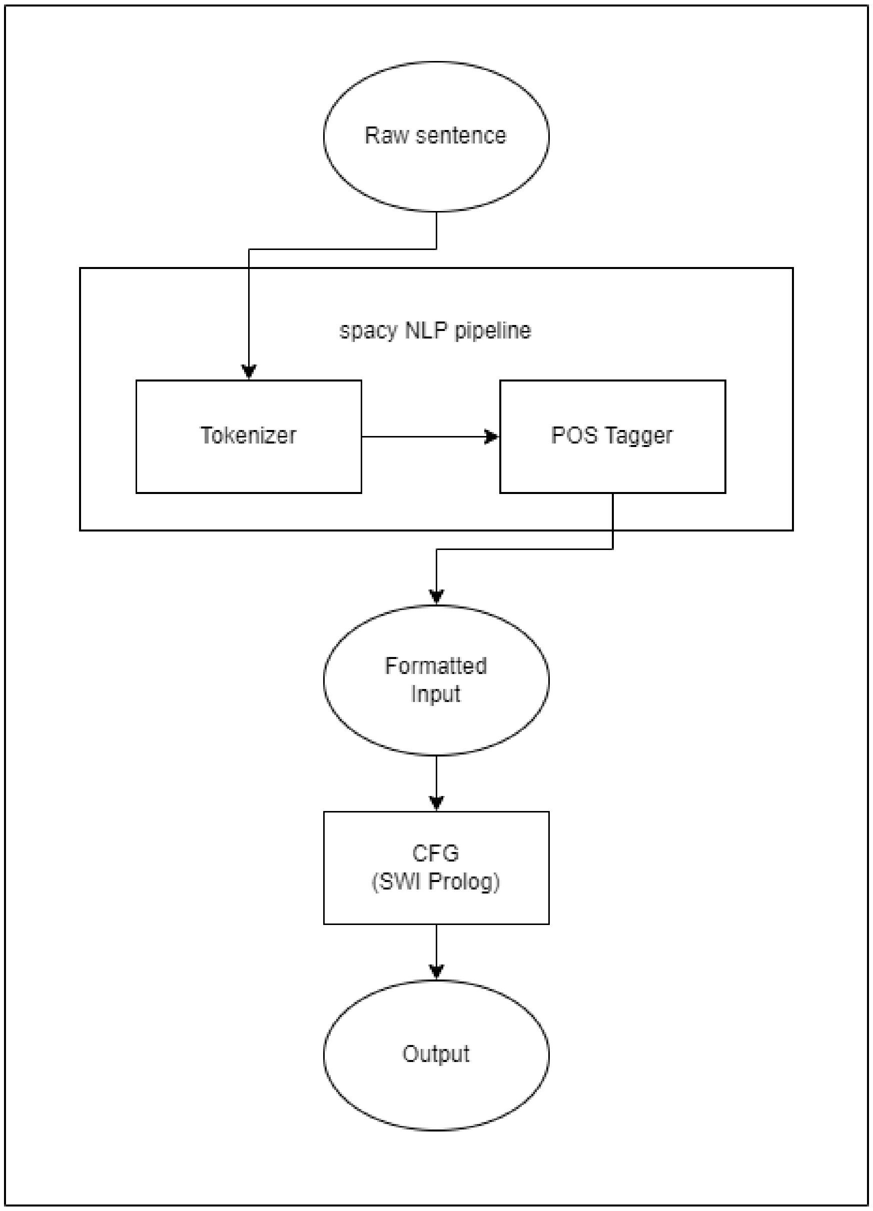


Figure 3.1: Data Pipeline

Figure 3.1 shows the stages of processing involved in the transformation of unstructured data into structured RDF triples.

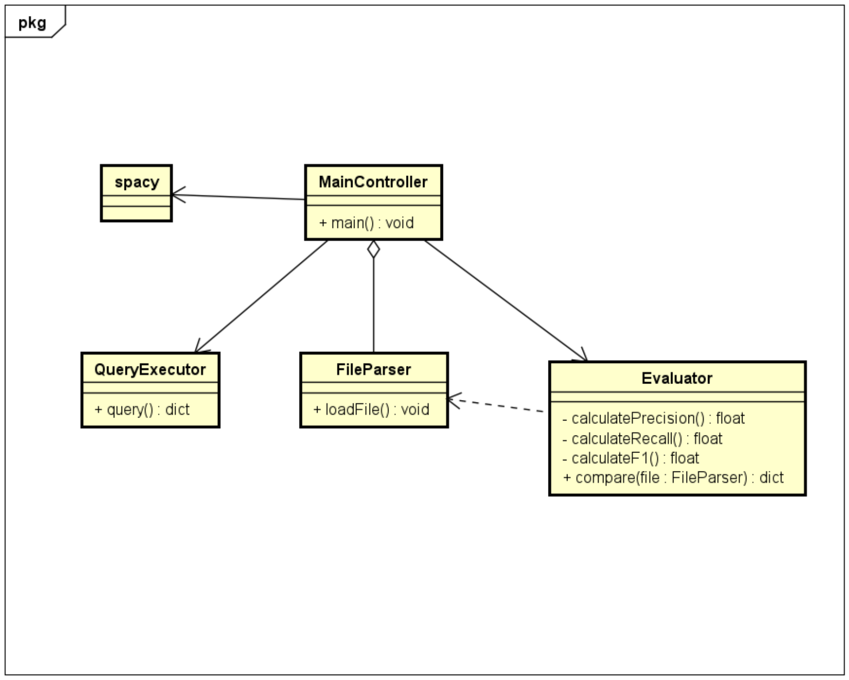


Figure 3.2: Class Diagram of the Application

Figure 3.2 shows the high-level architecture of the application used to conduct this study, including the task of open information extraction (OIE) as well as the evaluation of our proposed context-free grammar (CFG) based approach.

Chapter 4

DATA COLLECTION AND PROCESSING

4.1 Data Sources

4.1.1 Datasets for Evaluation

This chapter describes the datasets used for evaluation in this research study. The following sections further describe the datasets and pre-processing done on them.

For this thesis, we have considered the following datasets:

- the ReVerb dataset, which consists of 500 sentences randomly sampled from the web, using the Yahoo random-link service, by the authors of ReVerb system. This dataset was used in the study (22). These sentences are a bit noisy and their corresponding triples were manually generated for evaluation.

Sentence
The nation 's health maintenance organizations were required to tell the federal government by midnight Monday whether they plan to continue providing health insurance to Medicare recipients next year , raise premiums , or reduce benefits .
A backdrop of steady economic growth , scant inflation , and stable bond yields provided support for stocks .
E-mail : jsuydamstatesman.com In person Ms. Holloman looks less like a tomboy than her film roles suggest , with her hazel eyes and faint drawl her most beguiling features .
By the early 1980s , it was ZZZZ Best that showed how easy fraud could be .
E-mail her at mmccartycoxohio.com A Ply Gem Industries Inc. shareholder is suing Furman Selz Inc. for $ 100 million , claiming it was negligent in recommending Nortek Inc. 's takeover offer as fair to shareholders .

Table 4.1: Example Sentences from ReVerb Dataset

- WIKI dataset (21): 200 sentences were randomly sampled from Wikipedia documents by the authors of ClausIE and ground truth was manually generated. These sentences are shorter, simpler, and less noisy in comparison with the other datasets. Also, some Wikipedia articles are written by non-native speakers, however, the Wikipedia sentences do contain some incorrect grammatical constructions thereby posing some challenges to the extractions.

Sentence
Henry was Governor Edwin Washington Edwards's choice for Speaker.
Mrs. Yogeswaran was shot five times with a pistol near her Jaffna home on May 17, 1998.
Girard has a bachelor's degree in political sciences at the Universite de Montreal and did studies for the master's degree in industrial relations.
It was #109 on the Billboard 200 chart.
Robert Barnard (born 23 November 1936) is an English crime writer, critic and lecturer.

Table 4.2: Example Sentences from WIKI Dataset

- NYT dataset (36): New York Times annotated corpus has numerous news articles published by the New York times. This dataset comprises of 200 sentences randomly samples from this corpus by the authors of ClauseIE. These sentences were generally clean, however, tend to be long and complex. They manually generated the ground truth, that is the extracted triples for evaluation.

Sentence
CourtLink, which has 160 employees, developed a filing system that enables judges, lawyers and court clerks to process pleadings, motions and other documents electronically over a secure connection.
A graduate of the University of Arizona, she is a lecturer in writing at Princeton University and an author under the name of Kathryn Watterson Burkhart.
The events occurred in July.
Terrorist attacks by E.T.A. have declined in recent years and the number of its hardcore militants is thought to have fallen from the hundreds of 15 years ago to several score.
Mr. Mahdi and Mr. Chalabi say they aim to form a ”national unity” government with Iraq’s main political leaders, presumably including Mr. Allawi.

Table 4.3: Example Sentences from NYT Dataset

With these datasets, we aim to create a comprehensive sample space that is representative of the varied styles and complexity of the English language. The Wikipedia (WIKI) and New York Times (NYT) datasets together create a spectrum of sentences ranging from simple assertions to complex structures interspersed with punctuations and nested clauses. However, the ReVerb dataset is generally noisy and the sentences frequently contain both grammar and spelling errors. For this reason, we have chosen to omit this dataset.

4.2 Pre-Processing

Since both the WIKI and NYT datasets are generally clean, we did not need to remove any sentences or change them in any way. First, we removed any extractions that are not valid strictly as a triple. Since ClausIE can make extractions with a degree greater than or less than 3, we exclude those extractions. Within the scope of this study, we are only concerned with triple extractions.

Next, we proceed with the natural language processing (NLP) operations.

Using the spacy library, we tokenized each sentence and assigned part-of-speech (POS) tags to each token. The inputs for the CFG model are stored as a list of tuples, where each tuple represents a single token from the sentence along with its POS tag. Furthermore, each of the correct triple extractions in the dataset corresponding to the particular sentence were compiled for the evaluation and result analysis as described in the next chapter..

Chapter 5

EVALUATION AND RESULTS

5.1 Experimental Setup

We will evaluate the performance of our proposed CFG model on the NYT and WIKI datasets. Each sentence is passed through the spacy NLP pipeline (tokenizer + POS tagger) as part of the pre-processing phase. Next, the formatted input is passed to the Pyton controller which uses pyswip to pass a query using the rules of the grammar written in SWI Prolog. Once the triples are generated, these are validated against the correct extractions. The dataset includes the triples extracted by ClausIE, ReVerb and OLLIE whose results will be compared with those of our proposed model that combines learning and rules using the CFG model.

5.2 Discussion

Before we look at the performance metrics, it is important to note that the dataset only provides correct/incorrect labels for triples generated by other models. In this section, let us examine some triples generated by the proposed CFG model that have not been classified by the existing prior systems.

First, we have the sentences that contain quotes and implications. The CFG model can effectively extract the subject clause with a predicate like 'say', 'claim' etc. with the intended quote or statement. Furthermore, we can also treat the quote or implied statement as a separate sentence on its own to extract more information. Table 5.1 below shows examples of such sentences with quotations and implied statements that our proposed model is able to handle.

Table 5.1: Quotations and Implied Statements

Reference	Sentence	Triple
nyt-4	Mr. Mahdi and Mr. Chalabi say they aim to form a ”national unity” government with Iraq's main political leaders, presumably including Mr. Allawi.	('Mr. Mahdi and Mr. Chalabi', 'say', 'they aim to form a ”national unity” government with Iraq's main political leaders')
nyt-25	McGaughey said that Personal Ensign might race again in New York this year and definitely would race next year and be pointed for the Breeders' Cup.	('McGaughey', 'said', 'Personal Ensign might race again in New York this year and definitely would race next year and be pointed for the Breeders' Cup')
nyt-46	He said the public school from which he transferred three years ago had metal detectors that still failed to keep out the sense that danger lurked nearby.	('He', 'said', 'the public school from which he transferred three years ago had metal detectors that still failed to keep out the sense that danger lurked nearby')
		Continued on next page

Table 5.1

Reference	Sentence	Triple
nyt-55	A defense spokesman added that British officials were "aware of the potential" for missile attacks along the border between the province and the Irish republic.	('A defense spokesman', 'added', 'British officials were " aware of the potential " for missile attacks along the border between the province and the Irish republic')
nyt-108	Sharpe said the students were sent to work in the manure, not stand in it.	('Sharpe', 'said', 'the students were sent to work in the manure , not stand in it')
nyt-120	The International Business Machines Corporation said it signed an agreement to provide more than $100 million worth of computer hardware and software to what will be the world's largest computerized travel reservation system.	('The International Business Machines Corporation', 'said', 'it signed an agreement to provide more than $ 100 million worth of computer hardware and software to what will be the world's largest computerized travel reservation system')

Continued on next page

Table 5.1

Reference	Sentence	Triple
nyt-86	Indeed, history shows that those who ignore energy prices and their impact on overall inflation do so at their own risk.	('history', 'shows', 'those who ignore energy prices and their impact on overall inflation do so at their own risk')

Next, we will look at sentences with complex syntactic structures in which the subject, predicate, and object clauses rarely occur in a linear fashion. In these cases, we can see the strengths of the proposed CFG approach in being able to match the right clauses together. This is achieved using rules with nested sentences, and matching a subject or object clause of the high-level sentence with the other clauses belonging to a lower-level or nested sentence. Table 5.2 shows examples of such sentences with nested or embedded clauses that the proposed approach can handle.

Table 5.2: Sentences with Nested/Embedded Clauses

Reference	Sentence	Triple
nyt-1	A graduate of the University of Arizona, she is a lecturer in writing at Princeton University and an author under the name of Kathryn Watterson Burkhart.	('she', 'is', 'A graduate of the University of Arizona')

Continued on next page

Table 5.2

Reference	Sentence	Triple
nyt-11	In its most recent survey, the Congress for New Urbanism, a non-profit organization based in Chicago, reported 648 neighborhood-scale New Urbanist communities in the United States, an increase of 176 over a 12-month period.	('the Congress for New Urbanism', 'based in', 'Chicago')
nyt-47	Nordstrom Inc., the retail chain based on the West Coast, said today that it had created an office of the president to strengthen its executive management team.	('Nordstrom Inc.', 'based on', 'the West Coast')
nyt-51	In times past, getting a taxi at a New York City airport was often a Darwinian affair, a frenzied, survival-of-the-fittest scene of piled-up luggage, darting taxis and shouting cabbies looking for an edge.	('darting taxis and shouting cabbies', 'looking for', 'an edge')

Continued on next page

Table 5.2

Reference	Sentence	Triple
nyt-107	In a sobering speech meant to register with opinion leaders in the international community, the Irish Foreign Minister, Dick Spring, has warned about the peril to the peace process if there is further delay.	('Dick Spring', 'has warned about', 'the peril to the peace process if there is further delay')
nyt-175	Divertimento" from "Le Baiser de la Fee," presented by the New York City Ballet on Wednesday night at the New York State Theater, is one of George Balanchine's odder ballets.	('Divertimento', 'is', "one of George Balanchine's odder ballets")
wiki-36	The Islamic Jihad Union (IJU), also known as Islamic Jihad Group (IJG), is a terrorist organization which splintered from the Islamic Movement of Uzbekistan (IMU), and has conducted attacks in Uzbekistan and attempted attacks in Germany.	('The Islamic Jihad Union', 'is', 'a terrorist organization')

Continued on next page

Table 5.2

Reference	Sentence	Triple
nyt-94	Russia's eavesdropping station at Lourdes, Cuba, is a cold war jewel: 28 square miles of antennae and computers that message Russian embassies, submarines and spies, and vacuum up U.S. satellite and radio communications.	("Russia's eavesdropping station at Lourdes", 'is', 'a cold war jewel')
nyt-106	Trust Company's quest for customers among the new rich, included erroneous figures for millionaire households supplied by PSI, a financial services research firm in Tampa, Fla.	("Trust Company's quest for customers", 'included', 'erroneous figures')

5.3 Experimental Results

We evaluate our proposed approach by comparing our proposed CFG model with other existing systems - ClausIE, ReVerb, and OLLIE. The evaluation metrics used are as follows:

- Number of correct triples extracted

- Precision: is the fraction of the returned extractions that are correct.

- Recall: is the fraction of correct extractions in the corpus that are returned.

- F1 score: is a combined score of the precision and recall calculated using the following formula

F1-score = 2 * (precision * recall) / (precision + recall)

	ClausIE	ReVerb	OLLIE	CFG
Wiki	598	165	234	128
NYT	696	149	211	158

Table 5.3: Correct Triples Extracted

	ClausIE	ReVerb	OLLIE	DeepEx	CFG
Wiki	0.597	0.663	0.414	—	0.406
NYT	0.534	0.550	0.425	0.815	0.434

Table 5.4: Precision

	ClausIE	ReVerb	OLLIE	DeepEx	CFG
Wiki	0.644	0.178	0.252	—	0.138
NYT	0.682	0.146	0.207	0.899	0.155

Table 5.5: Recall

	ClausIE	ReVerb	OLLIE	DeepEx	CFG
Wiki	0.620	0.281	0.313	—	0.206
NYT	0.599	0.231	0.278	0.855	0.228

Table 5.6: F1 Score

Chapter 6

FUTURE WORK

6.1 Error Analysis

Let us start by analyzing the output of the CFG model, both for cases where it generated no outputs and incorrect outputs. In this section, we will categorize and present the main error types and their causes, and we will discuss potential solutions to fix or mitigate these problems.

6.1.1 Semantics

Reference	Sentence	Triple Generated	Actual Triple
nyt-168	She was a continuing source of love, kindness and generosity who never failed to nurture her adoring family.	('She', 'was', 'a continuing source')	('She', 'was', 'a continuing source of love')
wiki-76	76 In 2009, Hobbs acquired the role of top gymnast, Emily Kmetko, in ABC Family's "Make It or Break It".	('Hobbs', 'acquired', 'the role')	('Hobbs', 'acquired', 'the role of top gymnast')

Table 6.1: Semantic Errors

Depending on the semantics of the predicate and the object entity, the object clause may or may not be extended by prepositions or adpositions. In Table 6.1, we can see some examples of this behaviour. However, this occurence of extended object clauses using an adposition or a preposition does not always occur. Even with the same POS tags, the triple formation can be different for different sentences due to the semantics of the object or predicate phrase. This is a limitation of the CFG model as it cannot account for different rules based on the semantics of the text.

6.1.2 Compound Predicates

Reference	Sentence
wiki-7	It comprises and includes mountains reaching an altitude of above sea level.
wiki-51	While the Arab world is a rich prize in itself, Europe has been and remains the primary objective.
nyt-199	General Blevins agreed and said he would strike with a tank regiment held in reserve as well as with helicopters and Air Force support.

Table 6.2: Sentences with Complex Predicates

Compound predicates are usually formed by a combination of simple predicates using coordinating conjunctions. In Table 6.2, we can look at a few examples of this case. It appears that the best solution would be to break down the compound predicate into its individual atoms, and then create a triple for each atomic predicate with the same subject and object clause, but this may not always work.

Looking at the sentence wiki-7, ”comprises and includes” can be broken down into ”comprises” and ”includes”, and two triples can be formed using these atomic predicates separately which are as follows:

- (”It”, ”comprises”, ”mountains reaching an altitude of above sea level”)
- (”It”, ”includes”, ”mountains reaching an altitude of above sea level”)

However, this does not work for the sentence nyt-199. The compound predicate ”agreed and said” can be broken down into ”agreed” and ”said”. But the triple can only be formed with ”said” and not with ”agreed”. This is because ”agreed” does not have anything to do with the object clause and thus cannot be used to form a triple. We need to create more specific rules to deal with different types of compound predicates. Another key idea is to create specific rules for every individual predicate that can be further specialized with the advantage of knowing its semantic usage and context.

6.1.3 Special Characters

Reference	Sentence
wiki-56	A Great-Great Grandson was Robert Norton Noyce (1927 - 1990), nicknamed ”the Mayor of Silicon Valley”, co-founder of Fairchild Semiconductor in 1957 and Intel in 1968.
nyt-14	But inexpensive point-and-shoot cameras can do the job if they have a telephoto setting or a zoom lens.
nyt-109	Shown today at 2 and 6 p.m. and tomorrow at 4 and 7 p.m. at the Walter Reade Theater at Lincoln Center, 165 West 65th Street, Manhattan, as part of the 32nd New Directors/New Films series of the Film Society of Lincoln Center and the department of film and media of the Museum of Modern Art.

Table 6.3: Sentences with Special Characters

The usage of special characters can cause problems in the tokenization phase leading to the CFG rules failing to parse the provided input. Let us look at a few examples in Table 6.3. In each of these sentences, the tokenizer treats ’-’ or ’/’ as a separate token. This causes problems since we expect ”Great-Great” or ”point-and-shoot” to behave as a single adjective token. Moreover, in sentece nyt-109, the ’/’ symbol is actually joining ”New Directors” and ”New films”. These are some of the different kinds of behavior caused by special characters that need to be accounted for.

6.1.4 Lack of entities or Relations

Reference	Sentence
nyt-102	Beat in the eggs one at a time.
nyt-30	To reach the museums by foot, leave from the station's 30th Street exit and walk along J.
wiki-163	Great if the president will help but totally unnecessary
wiki-14	Simultaneously won the ”Hope of the World Ballet” Prize.

Table 6.4: Sentences Lacking Entities or Relationships

Some sentences are generally not suitable for the task of open information extraction. Either they do not have well defined entities, or they are imperative or exclamatory sentences that do not contain any patterns of the subject-predicate-object format that is necessary for forming triples. In Table 6.4, we can look at a few examples of sentences that belong in this category. It is hard to form any meaningful triples with such sentences and they should be excluded from the dataset in the preprocessing phase since they do not fall under the scope of this task.

6.2 Improvements

In terms of future work, these are some of the features that fall under the natural progression of the CFG model that would increase its effectiveness and versatility.

- **Coreference resolution**

  Coreference resolution is defined as the task of matching any given pronoun in the natural language data sample with the noun phrase or entity that it references. It does not come under the scope of this thesis because the datasets we used contain single sentences randomly sampled from larger documents, which often means that a pronoun used in the given sentence may very well be referring to an entity that belongs to its parent document but does not exist in the given sentence. However, when we eventually extend the scope to paragraphs or even entire documents, this would be an essential feature to further increase the recall of the CFG model.

- **Extracting information from subclauses**

  Many sentences contain information that can be extracted as triples from their subclasses or nested clauses. Usually, these occur as quoted statements, or clauses containing possessive particles, or implied punctuations. For example,
  "My book is red."
  forms the basic triple
  ("My book", "is", "red")
  However, if we further attempt to parse the subject clause, we can get another triple
  ("I", "HAS", "book")
  Let us consider another example - "Steve, a gardener, likes apples".
  If we closely look at the first part of the sentence "Steve, a gardener, ...", we can extract a triple of the form
  ("Steve", "IS", "a gardener")
  Adding features to support this capability will significantly increase the amount of information extracted from a single sentence thereby increasing the overall performance of the CFG model.

- **Defining rules for specific entities**

  As observed in the previous section, the CFG model is prone to errors arising from the fact that many predicate clauses behave differently under different semantic context. To address this issue, we propose to reduce the scope of predicate rules by creating more specialized rules for different verb phrases based on their usage and context. This way, we could increase the accuracy of the extracted triples containing the given predicate. Moreover, the CFG model may not always be used for open information extraction, and for specific use cases, limiting the allowed predicates and adding support with highly specific rulesets would greatly increase the precision of the CFG model.

6.3 Context-Focused Extraction

Based on our analysis of the model, we can see that due to semantic errors within object clauses and compound predicates, the CFG model does not perform at par with the conventional ML models. However, it shows great promise as it does address many of the concerns we stated in our hypothesis and we propose a more focused and context-specific approach.

Adding context to a context-free grammar rule-based model seems counterintuitive. However, based on the error analysis in Chapter 5, it is evident that phrases from natural language form different triples even with the same POS tag syntax.

We propose that we inject semantics into our CFG rules by adding a third parameter - the actual word itself. By introducing a lemmatizer into the NLP pipeline, we can change our basic input into the format
<token, POS-tag, lemma>

Next, we can write new rule(s) specifically aimed at parsing the given occurrence of the word in the subject, predicate, or object phrase. This can allow our CFG model to behave differently when these special cases occur.

We could do this from scratch, but we already have some rules for parsing predicates so we could tweak those to suit our needs. By using the extra information from the lemma form of the token, we can make the rules more specialized to the cases dealing with the mentioned words only. Moreover, as part of Prolog functionality, these rules which are more specific can be placed at a higher priority than the generic rules so the extracted triples are based on the aforementioned specific rules we define.

Chapter 7

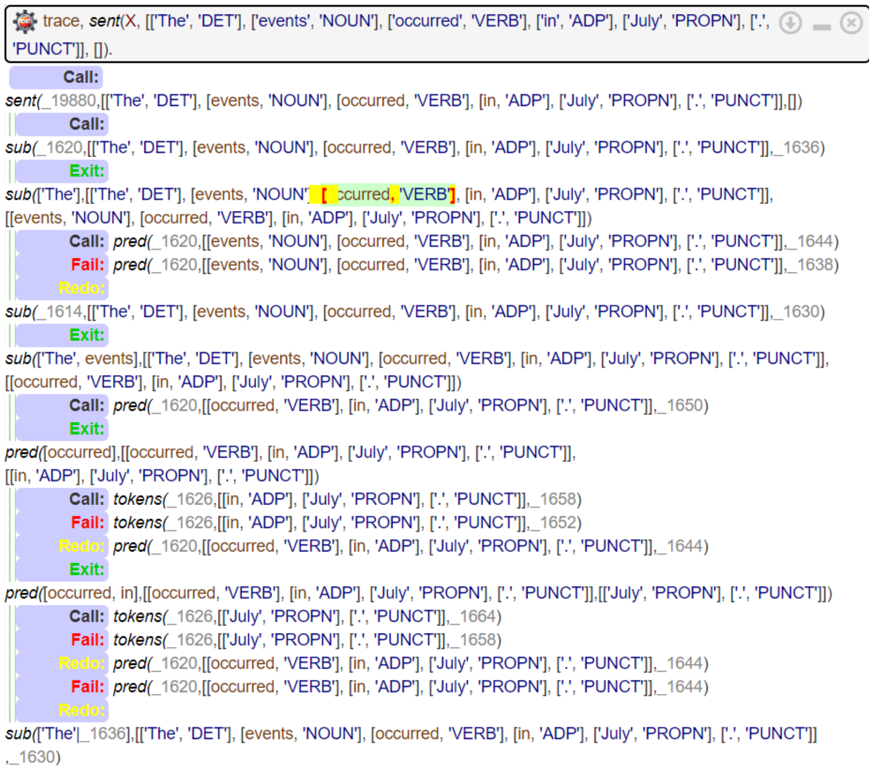
CONCLUSION

7.1 Our Contribution

In this thesis, we extended the domain of open information extraction with a novel approach using context-free grammar. In contrast with conventional machine learning models, our CFG model does not need any training data - as long as we have a firm grasp on the grammar of a language, we are able to write a set of rules in Prolog that can effectively parse it and generate meaningful triples. Moreover, this also increases the adaptability of our CFG model since major changes can be made by simply editing and changing the grammar rules, without the need for any new training datasets or cumbersome computations to re-train a neural network. In theory, this approach can also be extended to other languages. However, it is important to note that this is highly dependent on the existence of part-of-speech taggers for the given language. Lastly, we feel that one of the greatest strengths of our CFG model is transparency. With deep and complex machine learning models becoming the norm in most domains across the board, our CFG model guarantees transparency and accountability, since we have the ability to analyze the exact sequence of Prolog rules used to generate a particular output. This greatly increases both the accountability and reliability of our CFG model.

7.2 Hypothesis

We have achieved our goals and addressed the problems we wanted to address as part of our hypothesis.



```
trace, sent(X, [['The', 'DET'], ['events', 'NOUN'], ['occurred', 'VERB'], ['in', 'ADP'], ['July', 'PROPN'], ['.',
'PUNCT']], []).
     Call:
sent(_19880,[['The', 'DET'], [events, 'NOUN'], [occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],[])
      Call:
sub(_1620,[['The', 'DET'], [events, 'NOUN'], [occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],_1636)
      Exit:
sub(['The'],[['The', 'DET'], [events, 'NOUN'], [ ccurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],
[[events, 'NOUN'], [occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']])
      Call: pred(_1620,[[events, 'NOUN'], [occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],_1644)
      Fail: pred(_1620,[[events, 'NOUN'], [occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],_1638)
      Redo:
sub(_1614,[['The', 'DET'], [events, 'NOUN'], [occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],_1630)
      Exit:
sub(['The', events],[['The', 'DET'], [events, 'NOUN'], [occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],
[[occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']])
      Call: pred(_1620,[[occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],_1650)
      Exit:
pred([occurred],[[occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],
[[in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']])
      Call: tokens(_1626,[[in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],_1658)
      Fail: tokens(_1626,[[in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],_1652)
      Redo: pred(_1620,[[occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],_1644)
      Exit:
pred([occurred, in],[[occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],[['July', 'PROPN'], ['.', 'PUNCT']])
      Call: tokens(_1626,[['July', 'PROPN'], ['.', 'PUNCT']],_1664)
      Fail: tokens(_1626,[['July', 'PROPN'], ['.', 'PUNCT']],_1658)
      Redo: pred(_1620,[[occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],_1644)
      Fail: pred(_1620,[[occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']],_1644)
      Redo:
sub(['The'|_1636],[['The', 'DET'], [events, 'NOUN'], [occurred, 'VERB'], [in, 'ADP'], ['July', 'PROPN'], ['.', 'PUNCT']]
,_1630)
```

Figure 7.1: Output of Trace Command in Prolog

- **Training Data**

  Our CFG approach does not require any training data since there is no training process for a rule-based system using Prolog DCG rules to parse natural language.

- **Customization**

  We have provided support for users to add their own rules into the Prolog DCG, or modify existing rules already present. Users can also leverage the prioritization of rules in Prolog, and place their custom rules at a higher priority than the base rules.

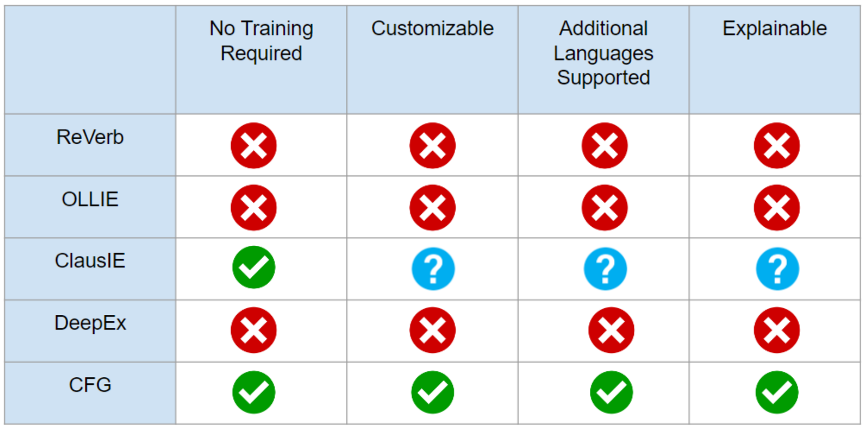
- **Additional Languages**

  Our CFG approach can be extended to other languages as well, given that the language has the concept of POS tags, and there exists a reliable POS tagger for the same. Additionally, one would need to be proficient in the language to write a comprehensive set of rules in Prolog.

- **Explainability**

  Using the trace command in Prolog as shown in figure 7.2, we can generate an audit log of the sequence of rules evaluated to generate our output. This makes our CFG approach completely transparent and explainable compared to other conventional machine learning models.

7.3 Summary



	No Training Required	Customizable	Additional Languages Supported	Explainable
ReVerb	✗	✗	✗	✗
OLLIE	✗	✗	✗	✗
ClausIE	✓	?	?	?
DeepEx	✗	✗	✗	✗
CFG	✓	✓	✓	✓

Figure 7.2: Summary of Our Hypothesis

We aimed to create a context-free grammar for the task of open information extraction. We created a set of rules in Prolog that work to parse natural language into triples, and we tested our model on datasets sourced from Wikipedia and the New York Times. Looking back at our hypothesis, we have managed to address the problems including but not limited to adaptability, reliability, and transparency. However, given the fact that our CFG model does not perform at par with the other conventional machine learning approaches, we believe that our hypothesis is only partially correct - it is hard to effectively account for the variations in semantics in natural language. However, we aim to address this in the future by customizing rules to be domain-specific and even predicate-specific in some cases. While we conclude that our hypothesis is partially correct, we have created a novel approach that sets the foundation for future attempts to build and improve upon.

7.4 Vision for Future Application

Our high-level vision for the CFG model is a vast open-source repository, that contains rules aimed at various information extraction tasks, with different goals, and catering to various domains. This would be a dynamic database, but also a space for linguists, developers, and researchers to collaborate and share their knowledge and expertise by creating and optimizing the grammar rules. A repository of this kind provides a viable alternative to proprietary solutions or conventional machine learning methods that operate in a black box. Whether it be an individual or an organization, they can benefit equally from this repository regardless of size and access to resources. With this repository, we hope to encourage collaboration that leads to accelerated advancements and innovations in the field of information extraction and natural language processing (25).

REFERENCES

[1] M Banko, MJ Cafarella, S Soderland, M Broadhead, and O Etzioni. Open information extraction from the web in: Proceedings of the 20th international joint conference on artificial intelligence. 2007.

[2] Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. A survey on open information extraction. *arXiv preprint arXiv:1806.05599*, 2018.

[3] World Wide Web Consortium (W3C). Rdf 1.1 xml syntax. https://www.w3.org/standards/semanticweb/, 2014. [Online; accessed 06-June-2023].

[4] Sarthak Tiwari, Bharat Goel, and Srividya Bansal. Mold-a framework for entity extraction and summarization. In *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, pages 445–450. IEEE, 2020.

[5] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[6] Daniela Barreiro Claro, Marlo Souza, Clarissa Castellã Xavier, and Leandro Oliveira. Multilingual open information extraction: Challenges and opportunities. *Information*, 10(7), 2019.

[7] Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

[8] World Wide Web Consortium (W3C). Semantic web at w3c. https://www.w3.org/standards/semanticweb/, 2023. [Online; accessed 06-June-2023].

[9] The linked open data cloud. https://lod-cloud.net/. [Online; accessed 17-June-2023].

[10] Mark A Musen. The protégé project: a look back and a look forward. *AI matters*, 1(4):4–12, 2015.

[11] World Wide Web Consortium (W3C). Owl web ontology language overview, w3c recommendation, 10 feb. 2004. https://www.w3.org/TR/owl-features/, 2014. [Online; accessed 06-June-2023].

[12] Universal Dependencies contributors. Universal pos tags. https://universaldependencies.org/u/pos/universal-pos-tags, 2014. [Online; accessed 13-February-2023].

[13] William F Clocksin and Christopher S Mellish. *Programming in PROLOG*. Springer Science & Business Media, 2003.

[14] Python Software Foundation. About python. https://www.python.org/about/, 2012. [Online; accessed 15-April-2023].

[15] Yüce Tekol and PySwip contributors. About python. https://github.com/yuce/pyswip, 2007. [Online; accessed 15-April-2023].

[16] Raghu Anantharangachar, Srinivasan Ramani, and S Rajagopalan. Ontology guided information extraction from unstructured text. *arXiv preprint arXiv:1302.1335*, 2013.

[17] Farhad Abedini, Fariborz Mahmoudi, and Amir Hossein Jadidinejad. From text to knowledge: Semantic entity extraction using yago ontology. *International Journal of Machine Learning and Computing*, 1(2):113, 2011.

[18] Alain Auger and Caroline Barrière. Pattern-based approaches to semantic relation extraction: A state-of-the-art. *Terminology*, 14(1):1, 2008.

[19] Gerardo Sierra, Rodrigo Alarcón, César Aguilar, and Carme Bach. Definitional verbal patterns for semantic relation extraction. *Terminology. International Journal of Theoretical and Applied Issues in Specialized Communication*, 14(1):74–98, 2008.

[20] Guido Boella, Luigi Di Caro, and Livio Robaldo. Semantic relation extraction from legislative text using generalized syntactic dependencies and support vector machines. In *Theory, Practice, and Applications of Rules on the Web: 7th International Symposium, RuleML 2013, Seattle, WA, USA, July 11-13, 2013. Proceedings 7*, pages 218–225. Springer, 2013.

[21] Luciano Del Corro and Rainer Gemulla. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*, pages 355–366, 2013.

[22] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 1535–1545, 2011.

[23] Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg. Getting more out of syntax with props. *arXiv preprint arXiv:1603.01648*, 2016.

[24] Ronald Smith Djomkam Yotedje. Giet: Generic information extraction using triple store databases. *INFORMATIK 2015*, 2015.

[25] Michael Schmitz, Stephen Soderland, Robert Bart, Oren Etzioni, et al. Open language learning for information extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 523–534, 2012.

[26] Nikita Bhutani, HV Jagadish, and Dragomir Radev. Nested propositions in open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 55–64, 2016.

[27] Chenguang Wang, Xiao Liu, Zui Chen, Haoyun Hong, Jie Tang, and Dawn Song. Zero-shot information extraction as a unified text-to-triple translation. *arXiv preprint arXiv:2109.11171*, 2021.

[28] Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. T-rex: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[29] Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. A novel cascade binary tagging framework for relational triple extraction. *arXiv preprint arXiv:1909.03227*, 2019.

[30] Yubo Chen, Yunqi Zhang, and Yongfeng Huang. Learning reasoning patterns for relational triple extraction with mutual generation of text and graph. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1638–1647, 2022.

[31] Benfeng Xu, Quan Wang, Yajuan Lyu, Yabing Shi, Yong Zhu, Jie Gao, and Zhendong Mao. Emrel: Joint representation of entities and embedded relations for multi-triple extraction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 659–665, 2022.

[32] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017.

[33] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[34] Weischedel, Ralph, Palmer, Martha, Marcus, Mitchell, Hovy, Eduard, Pradhan, Sameer, Ramshaw, Lance, Xue, Nianwen, Taylor, Ann, Kaufman, Jeff, Franchini, Michelle, El-Bachouti, Mohammed, Belvin, Robert, and Houston, Ann. Ontonotes release 5.0, 2013.

[35] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[36] Sandhaus, Evan. The new york times annotated corpus, 2008.

APPENDIX A

PROLOG PARSER CODE

```
% Here we have the set of DCG rules that is used to parse a sentence based on its
% POS tags, and ultimately generate outputs as a triple
table sent/3, vp/3, sub/3, pred/3, obj/3.

% helper functions to enable parsing
takeAllButLast(List, Result):-removeLast(List, [], Result).
removeLast([_], R, R).
removeLast([H|T], Sol, Result):-append(Sol, [H], Sol2), removeLast(T, Sol2, Result).

% Basic rules for terminal symbols using nouns to form a subject
term0([X]) –> [[X, 'PROPN']].
term0([X, '.']) –> [[X, 'PROPN'], ['.', _]].
term0([X]) –> [[X, 'NOUN']].
term0([X]) –> [[X, 'NUM']].
term0([X]) –> [[X, 'ADJ']].
term0([X|S]) –> [[X, 'PROPN']], term0(S).
term0([X, '.'|S]) –> [[X, 'PROPN'], ['.',_]], term0(S).
term0([X|S]) –> [[X, 'NOUN']], term0(S).
term0([X|S]) –> [[X, 'NUM']], term0(S).
term0([X|S]) –> [[X, 'ADJ']], term0(S).

% adding support for pronouns and currency
term1([X]) –> [[X, 'PRON']]..
term1([X1, X2]) –> [[X1, 'SYM'], [X2, 'NUM']].
term1(S) –> term0(S).

% adding support for adjectives
term2([X|S]) –> [[X, 'ADJ']], term2(S).
term2([X|S]) –> [[X, 'VERB']], term2(S).
term2(S) –> term1(S).

% adding support for adverbs
term3([X|S]) –> [[X, 'ADV']], term2(S)..
term3(S) –> term2(S).

% adding support for determiners and quantity
term4([X]) –> [[X, 'DET']].
term4([X|S]) –> [[X, 'DET']], term3(S).
term4([X|S]) –> [[X, 'NUM']], term3(S).
term4(S) –> term3(S)..

% adding support for complex subjects formed using coordinating conjunctions term5(S)
–> term4(S1), [[X, 'CCONJ']], sub(S2), append(S1, [X|S2], S).
term5(S) –> term4(S).

% adding support for complex subjects formed using adpositions term6(S) –> term5(S1),
[[X, 'ADP']], sub(S2), append(S1, [X|S2], S).
```

```
term6(S) –> term5(S).

% the high level subject rule that encompasses all rules
% from term0 through term6 and uses them to parse a subject clause
sub(S) –> term6(S1), [[X, ’PART’]], sub(S2),
last(S1, L), atom_concat(L, X, Concat), takeAllButLast(S1, NewS1),
append(NewS1, [Concat], UpdatedS1),
append(UpdatedS1, S2, S).
sub(S) –> term6(S).
sub(S) –> [[’”’, ’PUNCT’]], tokens(Sub), [[’”’, ’PUNCT’]], append([’”’|Sub], [’”’], S).

% objects as simple terminals.
obj([X]) –> [[X, ’ADJ’]].
obj([X1, X2]) –> [[X1, ’ADV’], [X2, ’VERB’]].

% objects as extended subject clauses.
obj(X) –> sub(X).
obj(X) –> sub(S), [[V, ’VERB’]], append(S, [V], X).
obj([X1|S]) –> [[X1, ’ADP’]], sub(S).
obj(X) –> sub(X1), [[XADP, ’ADP’]], tokens(X2), append(X1, [XADP|X2], X).

% complex object rules.
obj(X) –> sub(X), [[_, ’ADP’]], tokens(_).
obj(X) –> sub(X), [[_, ’DET’]], tokens(_).
obj(X) –> sub(X), [[_, ’CCONJ’]], tokens(_).
obj(X) –> sub(X), [[_, ’SCONJ’]], tokens(_).
obj(X) –> sub(X), [[_, ’VERB’]], tokens(_).

% simple predicate rules using auxiliaries and verbs
pred([X]) –> [[X, ’VERB’]].
pred([X1, X2]) –> [[X1, ’ADV’] ,[X2, ’VERB’]].
pred([X1, X2]) –> [[X1, ’VERB’], [X2, ’ADP’]]..
pred([X]) –> [[X, ’AUX’]].

% more complex predicate clauses with support for additional constructs
pred([X1|X2]) –> [[X1, ’VERB’]], pred(X2).
pred([X1|X2]) –> [[X1, ’AUX’]], pred(X2).
pred([X0, X1|X2]) –> [[X0, ’VERB’], [X1, ’PART’]], pred(X2).
pred([X0, X1|X2]) –> [[X0, ’VERB’], [_, ’NOUN’] , [X1, ’PART’]], pred(X2).
pred([X0, X1|X2]) –> [[X0, ’AUX’], [X1, ’PART’]], pred(X2).

% n tokens.
tokens([X]) –> [[X, _]].
tokens([X|T]) –> [[X, _]], tokens(T).

% verb phrase sub-clause to reduce complexity of grammar
vp([P,O]) –> pred(P), obj(O).
vp([P,O]) –> pred(P), obj(O), [[’.’, ’PUNCT’]].
vp([P,O]) –> pred(P), obj(O), [[_, ’PUNCT’]], tokens(_).
vp([P,O]) –> tokens(_), [[_, ’PUNCT’]], vp([P,O]).

% high-level sentence rules that use the other sub-clause
% rulesets to parse a given sentence
sent([S,P,O]) –> sub(S), pred(P), tokens(O), [[’,’, ’PUNCT’]], tokens(_).
sent([S,P,O]) –> sub(S), pred(P), [[_, ’SCONJ’]], tokens(O), [[’.’, ’PUNCT’]].
sent([S,P,O]) –> sub(S), pred(P), [[_, ’NOUN’], [_, ’SCONJ’]], tokens(O), [[’.’, ’PUNCT’]].
sent([S,P,O]) –> sub(S), pred(P), tokens(O), [[’.’, ’PUNCT’]].

% sentence rules where S,P,O do not occur in a single sequence
% these rules deal with non-linear and nested clauses sent([S,P,O]) –> sub(S), vp([P,O]).
sent([S,P,O]) –> sub(S), [[’,’, ’PUNCT’]], sent([_,P,O]).
sent([S,P,O]) –> obj(O), [[’,’, ’PUNCT’]], sent([S,P]).
sent([S,P,O]) –> sub(S), [[’,’, ’PUNCT’]], vp([P,O]).
sent([S,P,O]) –> tokens(_), [[’,’, ’PUNCT’]], sent([S,P,O]).
sent(X) –> [[’”’, ’PUNCT’]], sent(X), [[’”’, ’PUNCT’]].
```

Available on GitHub at https://github.com/vsingh57/CFG_triple_extractor