Generation and Analysis of Semantic Metadata Records Based on the Targeted Retrieval of Open-Web Resources for an Automated Article Search Engine Agent

Project Overview

Introduction and Purpose

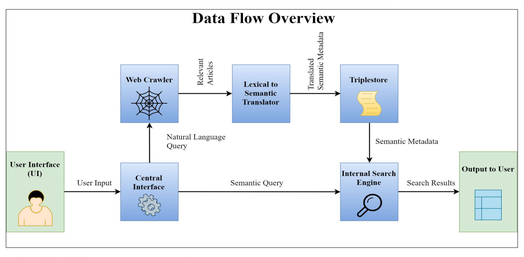
Here I introduce SHIRASE(Semantic-lexical Hybrid Information Retrieval And Search Engine). SHIRASE is a scientific article search engine designed to apply the versatility and specificity of a semantic search to access the breadth of data available on the conventional lexical web, thereby combining the best of both worlds. **This allows SHIRASE to provide a more fine-grained and nuanced search experience than conventional search engines.**

Standard search engines (Google, Bing, DuckDuckGo) are all at their core lexically based. This means that instead of being based around the meaning of the data they are processing, they are based around the words (syntactic structure) through which the information is conveyed. This means that **these search engines are inflexible in the types of outputs they can give, unable to interpolate information from multiple sources, and unable to handle complex queries.**

SHIRASE is able to "understand" the data it is processing by utilizing the formal semantics data format which explicitly represents the ideas of a given article in a network form. This allows SHIRASE to present a search solution which **directly answers user queries (not just output web pages which are relevant to the query), utilize data from multiple sources to formulate a response, and handle extremely advanced and complex search queries.**

In its current form, SHIRASE is designed as a knowledge search utility specifically for scientific literature, however **SHIRASE's architecture is inherently expandable and can eventually be made into a general search engine for the public.**

Overview of SHIRASE Pipeline



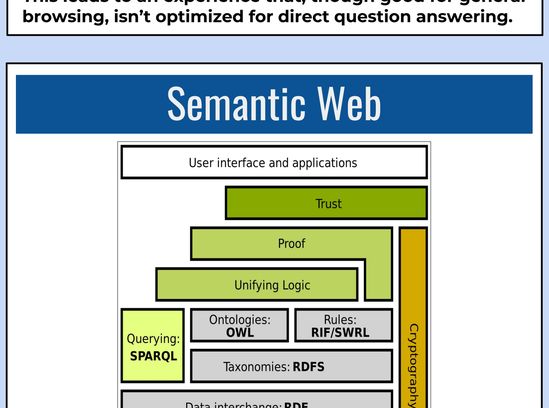
SHIRASE consists of 4 main sections: the web crawler, the lexical-to-semantic translator, the triplestore, and the internal search engine.

Web Crawler: The web crawler searches through various scientific literature databases for potentially relevant articles.

Lexical-to-Semantic Translator: The lexical-to-semantic translator extracts semantic markup of key ideas from the article text with a frame semantic parser.

Triplestore: The triplestore holds the RDF markup and citation metadata of the article in a remote or local location

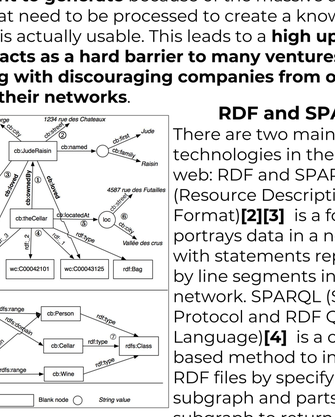
Internal Search Engine: The internal search engine utilizes SPARQL to search through the semantic records on the triplestore. This allows users to perform an in depth condition-based analysis of the SHIRASE database. SHIRASE also includes a query builder for ease-of-use.



Lexical Web

Currently, the internet is heavily lexically based. This means that it is based on the syntactic structure of words in a particular document rather than the meaning of the information being stored in the document itself. On an implementation level, this translates to data being stored on the web in such a way that computers, unless they use a form of metadata attached to an article (i.e. tags, categories, keywords, etc.), cannot understand the data that they are accessing. **Because of this, search systems have to depend on the lexical structure of the data to interpret the articles that the user wants to retrieve, thus leading to the prevalence of keyword matching search engines which attempt to match phrases or words from a user's query with key phrases or words** from a particular web page of interest. Though these can be effective in interpreting the intent of a user in many scenarios, it is nevertheless lacking in that the computer understands neither the information it is processing nor the query that the user has given. **Thus, current search engines are optimized to look for resources that may be relevant to a user's query rather than directly answering the query. This leads to an experience that, though good for general browsing, isn't optimized for direct question answering.**

Semantic Web



Semantic Web Introduction

The **semantic web[1] represents a paradigm shift in the way online data is stored and processed.** By weaning away from relying on the syntactic structure of the document and towards the explicit representation of the network of ideas in a resource, **the semantic web offers the capability for automated applications to "understand" the content they are processing**. Thus, it is far more intuitive to perform queries, since the ideas from a variety of resources are represented explicitly on a unified linked data graph. Effectively, the lexical web is container centric (based around web pages) while the semantic web is idea centric (based around the ideas contained within it regardless of the resource of origin). **This fact allows semantic applications to search for ideas and stand-alone answers, not just for web pages.**

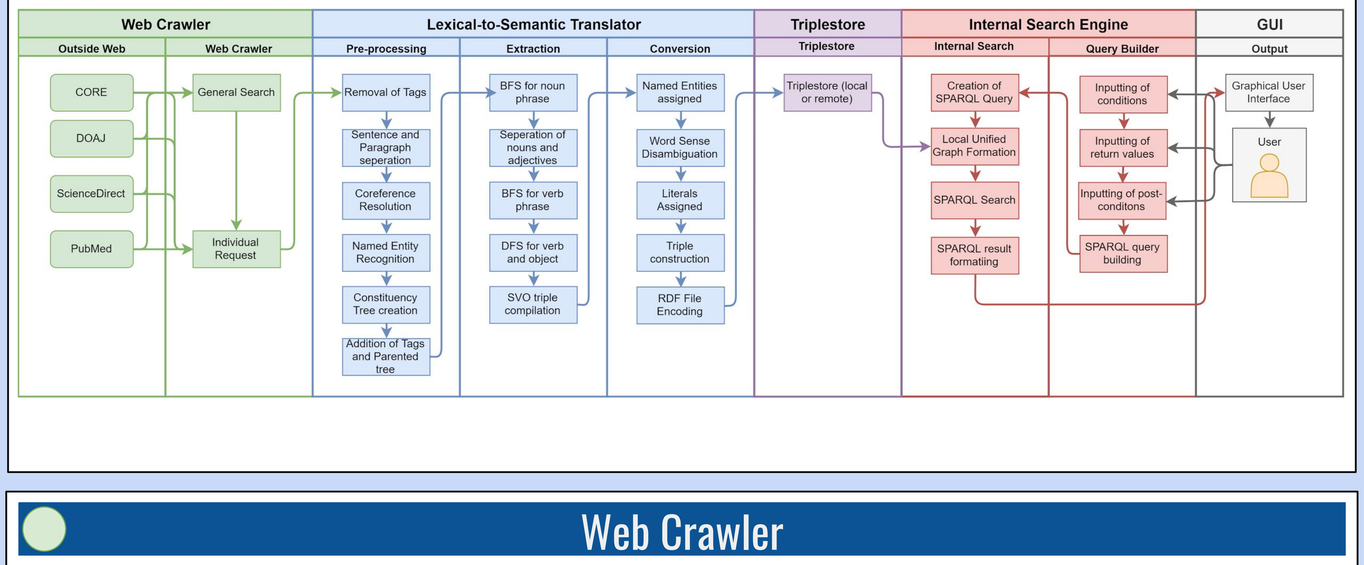
Even though the semantic web has so many advantages it has yet to hit the mainstream due to its few crippling drawbacks. The most pertinent of these is that most **semantic databases require a significant resource investment to generate** because of the massive amounts of data that need to be processed to create a knowledge base that is actually usable. This leads to a **high upfront cost that acts as a hard barrier to many ventures into the area along with discouraging companies from open-sourcing their networks**.

RDF and SPARQL

There are two main technologies in the semantic web: RDF and SPARQL. RDF (Resource Description Format)**[2][3]** is a format that portrays data in a network, with statements represented by line segments in the network. SPARQL (SPARQL Protocol and RDF Query Language)**[4]** is a condition-based method to interact with RDF files by specifying a subgraph and parts of the subgraph to return.

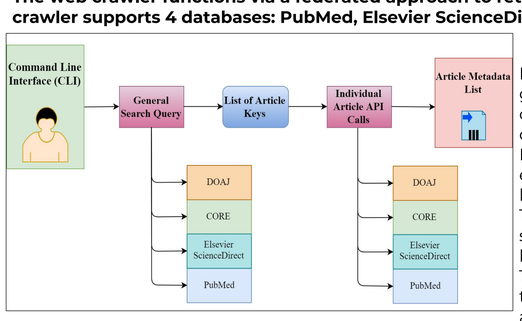
Methods and Components

SHIRASE Framework



Web Crawler

The web crawler functions via a federated approach to retrieve potentially relevant articles from various databases using REST API. Currently the web crawler supports 4 databases: PubMed, Elsevier ScienceDirect, CORE, and DOAJ. The web crawler has two main stages:



Step 1: General Search Query

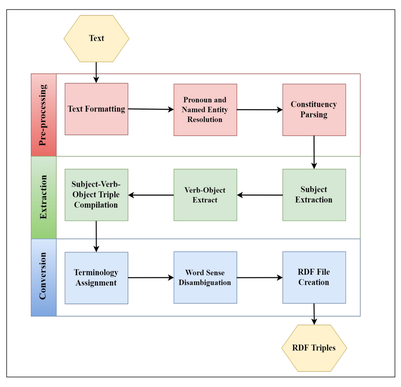
In the first step, the web crawler uses the general search query inputted by the user in order to search through the various literature databases. The web crawler does this by using REST API to access the inbuilt search utility for each database which returns a set of article keys to indicate potentially relevant articles. The general search query is in the form of a standard natural-language query that would be inputted into conventional search engines. The search engine then dynamically collates the results of each search to compile a list of article keys.

Step 2: Individual Article Requests

In the second step, the web crawler loops through the compiled list of article API keys and individually queries for each article to retrieve article metadata (author name, publication date, DOI, etc.) and text. Also, in this step the web crawler eliminates any articles that are not compatible (i.e. in a different language or in an irregular format). The text that is retrieved depends on whether or not the article's full paper is open access or not. If it isn't then just the abstract is retrieved since it most likely contains all the key claims.

Lexical-to-Semantic Translator

Purpose: The lexical-to-semantic translator takes text (of either the article or abstract) and extracts the statements made by the article based on discourse representation theory [5]. These effectively form a network which represents the ideas in a given article with each idea represented by a line segment.



Pre-processing

Text Formatting -

a) Raw Text Extraction: from the API result
b) Respacing: to correct for formatting
c) Paragraph Separation
d) Sentence Separation: via a rule learner

Entity Resolution -

a) Coreference Resolution: Statistical entity centric parser which uses a trained classifier and agglomerative clustering to prune mention pairs .
b) Named Entity Recognition: supervised learning linear chain Conditional Random Field (CRF) sequence model to discern long distance structures via Gibbs sampling to enhance probabilistic model.

Constituency Parsing - linear-time shift reduce constituency parser, which builds the constituency tree from the bottom-up using a series of transitions while correcting for grammar rules. Transitions predicted via a neural network classifier

Extraction

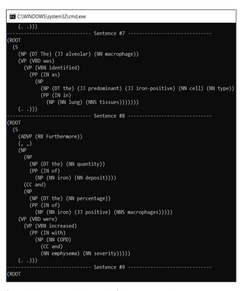
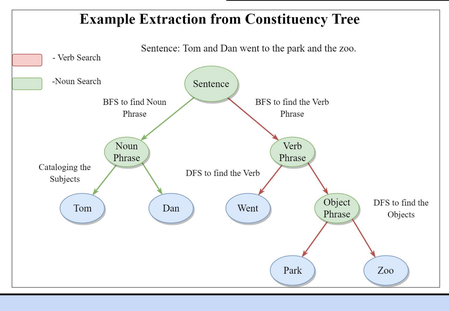


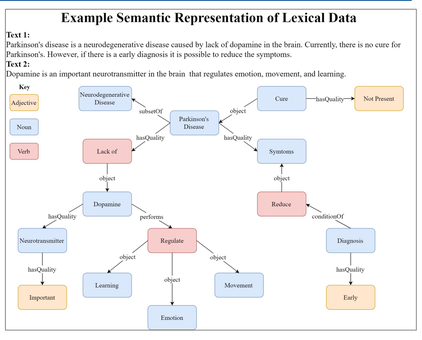
Fig. Desc.- Constituency trees extracted from the sentences.

First breadth-first search to find the noun phrase of the sentence. Then a two stage breadth-first and depth- first search is used to find the verb phrase and the predicate of the sentence. These are formed into triples.



Core Tree-Parsing Algorithm

```
SEM-EXTRACT(sentence)
    Triples ←
    SUBJ-EXTRACT (subtree-np)
    U PRED-EXTRACT(subtree-vp)
    U OBJ-EXTRACT(siblings-vp)
    if result ≠ false then return result
    else return false
SUBJ-EXTRACT(subtree-np)
    subject ← first found in NP_subtree
    subjectAttributes ←
    Attribute-Extract(subject)
    result ← subject U subjectAttributes
    if (result ≠ false): return result
    else: return false
PRED-EXTRACT(subtree-vp)
    predicate ← deepest verb in subtree
    predicateAttributes ←
    Attribute-Extract(predicate)
    result ← predicate U
    predicateAttributes
    if(result ≠ false): return result
    else: return false
OBJ-EXTRACT(subtree-vp)
    siblings ← find NP && PP && ADJP
    siblings of subtree-vp
    for value in siblings:
        if (value = NP || PP)
        object ← first noun value
        else:
        object ← first adjective value
    objectAttributes ←
    Attribute-Extract(object)
    result ← object U objectAttributes
    if (result ≠ false): return result
    else: return false
```



Conversion

Terminology Assignment: For the entities identified via the named entity recognition algorithm, they are assigned URI's (Unique Resource Identifiers) via various databases such as MeSH. These effectively allow for the universal identification of certain entities.

Word Sense Disambiguation: Performs graph compression in which words are assigned to synsets (grouping of words which are semantically equivalent) via Lesk. Lesk is a dictionary-based word sense disambiguation algorithm which operates using a supervised classifier that compares the context that an ambiguous word is placed in to other training texts. In addition, it also compares to words in semantic proximity of the prospect word allowing for greater accuracy.

RDF File Creation: Any nodes which are left (dates, unassigned proper nouns, numbers, etc.) are given XML datatypes. Then the triples are encoded into an RDF/XML file.

Result: The result of this process is a file composed of a series of triples. Each of these triples represents a line segment on the idea graph. This forms an explicit, formal representation of the ideas in an article which is machine-understandable.

Triplestore

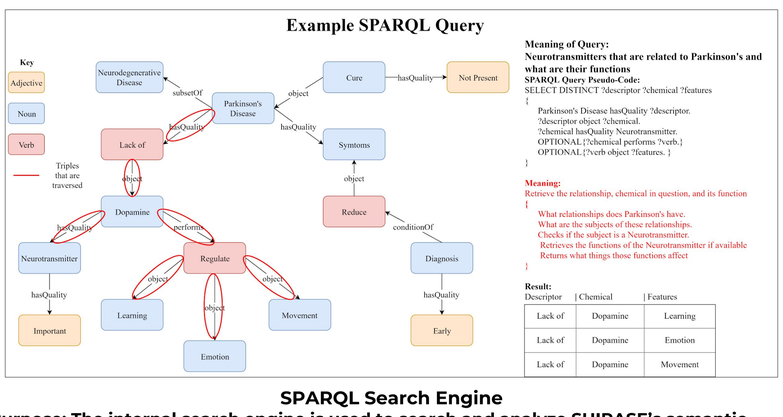
Triplestore Design

Once the RDF graph is created, it is stored in the triplestore. Inside the triplestore is a set of XML/RDF files, each of which describe an article. In regards to the structure of each RDF file, the graph is split into two sections: the citation metadata section and the text representation section. The citation metadata section holds basic information about the article (i.e. author name, date of publication, database of origin, etc.). the text representation section holds the triples from the lexical-to-semantic translator which create a network representation of the ideas in a particular article. The triple store can be stored either online on a server**[6]** via REST API and it can also be stored locally on the host computer.

Effect of Persistent Storage

Since the articles which are converted in a given search session are stored in the triplestore, it can be said that the information processed by SHIRASE is persistent. This has two effects. The first is that the articles processed and searched within one user's search can be used to augment subsequent searches and even another user's searches. It also leads to the gradual, targeted creation of a semantic database of converted articles which forms a knowledge graph. This process has the benefit of being distributed among many users over time, and also being driven by the users of SHIRASE, both in that their needs dictate what is added to the knowledge graph and in that they provide the computational resources for the conversion.

Internal Search Engine



SPARQL Search Engine

Purpose: The internal search engine is used to search and analyze SHIRASE's semantic database for information that the user has requested. Since it implements formal semantics, the internal search engine allows users to have specific parts of the retrieved articles returned to them without having to read through each article.

Design: This search engine is based primarily on SPARQL, however it does use some OWL-DL for basic logical reasoning and connection functions. The search engine operates by having a series of conditions which are inputted by the user to narrow down values of interest which can be returned. Using the conditions, the user can effectively designate a type of subgraph. Then they can further specify which parts of the subgraph they want returned to them. Since the graph represents the ideas of the article, users can effectively have specific ideas and information returned to them from the articles. In addition, through the use of predicate and first-order logic, the search engine can make simple deductions, and effectively create new connections in the graph.

Semantic Query Builder Form

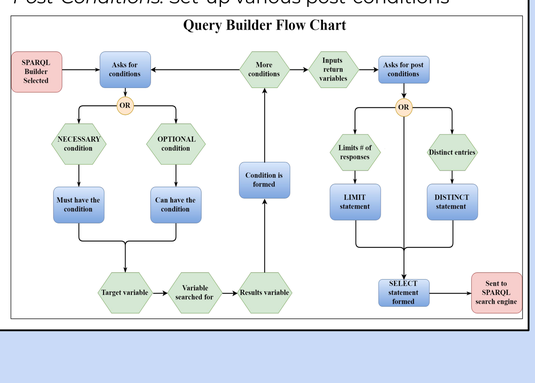
Purpose: Though the condition-based method of semantic search is extremely versatile and powerful, it isn't nearly as conducive as standard natural-language inputs. In an attempt to ameliorate this, SHIRASE implements an easy-to-use query builder to allows users to build semantic queries without having to know SPARQL syntax.

Design: The query builder consists of three segments:

Condition Input: Enter in the conditions for the query. The conditions are 3 parts with the first being target variable, the second being the condition, and the third being the result variable.

Output Choice: Select which variables are to be outputted

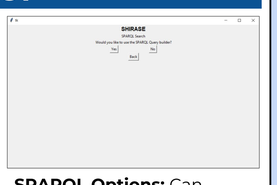
Post-Conditions: Set-up various post-conditions



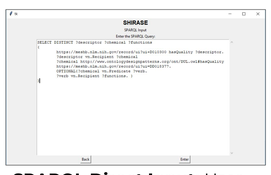
Results and Conclusions

GUI

Title Screen: Various search options are available for users to choose.

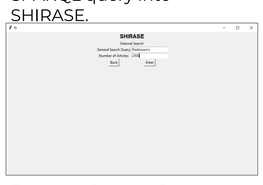


SPARQL Options: Can choose whether or not to use SPARQL builder form.



SPARQL Direct Input: User can directly input raw SPARQL query into SHIRASE.

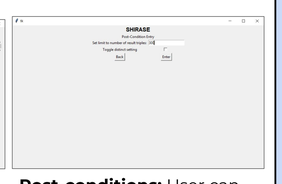
Condition Input: User inputs the conditions they want for the SPARQL query.



External Search: Search online for resources to add to the SHIRASE database.

Output Choice: User chooses output variables for the SPARQL query

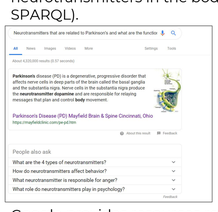
Output: SPARQL output given to a user with each column representing a variable.

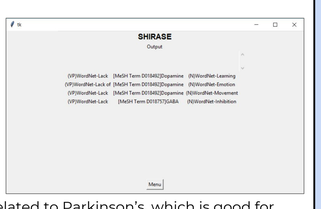


Post-conditions: User can input various post-conditions such as limiting # of results.

Results

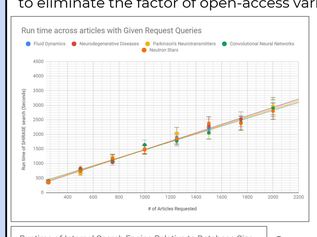
Example Query Comparison: Neurotransmitters that are related to Parkinson's and what are the function of those neurotransmitters in the body (for SHIRASE this is translated to SPARQL).



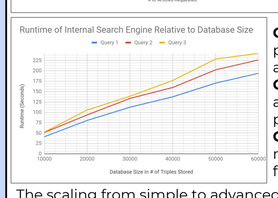


Google provides resources related to Parkinson's, which is good for browsing. However, SHIRASE provides a far more targeted and direct answer to the query by listing the neurotransmitters and functions.

Quantitative Performance Measures: This section portrays how SHIRASE performs as the scale of the requested query increases. The queries refer to the general search query inputted. The semantic query , which was kept constant, was limiting by publication date and returning the author/title. Note that only the abstracts are converted to eliminate the factor of open-access variability among articles.



The runtime seems to be relatively consistent across the board for all of the queries inputted. However, there seems to be a little variation, most likely due to the varying abstract sizes, though the total seems to average out. As a whole, it can be said that the runtime increases linearly with the request size.

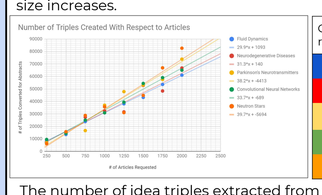


Query 1 (Simple): Limiting by publication date and returning title and author(s)

Query 2 (Intermediate): Extracting all of the functions that a certain predicate (verb) performs

Query 3 (Advanced): Cross-referencing citation networks to find which articles cited each other

The scaling from simple to advanced queries is relatively tame. A more minor note is that the runtime scaling levels out slightly as database size increases.



Query(Color represents query)	Avg. # of Triples / Article Abstract
	31.54
	33.39
	36.64
	32.54
	41.34

The number of idea triples extracted from the abstracts seem to be very variable, indicating that text size seems to be a significant factor in semantic metadata extraction.

Future Work and Conclusion

Future Work

Though SHIRASE is certainly effective, there are still numerous points of improvement:

Web Crawler: Needs to be moved away from relying on pure REST API infrastructure towards broader standards.

Lexical-to-Semantic Translator: The translator needs to have a shortened runtime, improved tree parsing), and novel summary statement formation.

SPARQL Search Engine: The SPARQL search engine can be improved by implementing a more advanced rule learner and a natural-language to SPARQL translator.

Conclusion

Given its advanced capabilities, SHIRASE can provide a more fine-grained and nuanced search experience than most lexically-based conventional search engines. SHIRASE does this by integrating the automated syphoning of lexical resources to augment user searches, **thus allowing for the versatility and specificity of a semantic search to be applied to resources on the conventional web. By integrating the breadth of the lexical web and versatility of the semantic web, SHIRASE is able to create a hybrid search engine.** This means that SHIRASE can bring semantics into the fore without succumbing to some of the weaknesses that have been holding the technology back. In addition, given its ability to process and convert large volumes of data from the open-web to its semantic database, **SHIRASE allows for the gradual, targeted creation of a semantic database driven by users for a variety of applications (i.e. plagiarism detection, text summarization, idea management)**. As a whole, **SHIRASE is a potent framework both as a search engine and for the distributed conversion of the lexical web to the semantic web.** Though at the moment SHIRASE is specifically for scientific literature, **the utility is built upon an expandable architecture that has the potential to become a generic search engine.**

Acknowledgements

I would like to acknowledge Dr. Carl Taswell and Adam Craig of Brain Health Alliance Virtual Institute (BHAVI) for their guidance.

Works Cited

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," Scientific american, vol. 284, no. 5, pp. 34–43, 2001.
[2] O. Lassila and R. R. Swick, "Resource description framework (rdf) model and syntax specification," 1999.
[3] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform resource identifier (uri): Generic syntax," Tech. Rep., 2004.
[4] S. Harris, A. Seaborne, and E. Prud'hommeaux, "Sparql 1.1 query language," W3C recommendation, vol. 21, no. 10, 2013.
[5] H. Kamp, J. Van Genabith, and U. Reyle, "Discourse representation theory," in Handbook of philosophical logic, Springer, 2011, pp. 125–394.
[6] C. Taswell, "A distributed infrastructure for metadata about metadata: The hdmm architectural style and portal-doors system," Future Internet, vol. 2, no. 2, pp. 156–189, 2010.