Tools for Linguistic and Semantic Resource Development

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in Computer Science and Engineering by Research

by

Grandhi Sai Venkata Harsha Vardhan
201002078
venkata.harshavardhan@research.iiit.ac.in

INTERNATIONAL INSTITUTE OF
INFORMATION TECHNOLOGY
HYDERABAD

International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2024

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled “Tools for Linguistic and Semantic Resource Development” by Grandhi Sai Venkata Harsha Vardhan, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Dr. Soma Paul

To my family

Acknowledgments

I want to express my sincere gratitude to everyone who has supported and guided me throughout this challenging journey of completing my thesis.

First and foremost, I owe a great debt of gratitude to my thesis advisor, Dr. Soma Paul. Their expertise, patience, and encouragement have been invaluable. Their insightful feedback and unwavering support have played a crucial role in shaping this work. Despite the challenges, their guidance helped me stay focused and motivated, ultimately enabling me to complete this research.

I am also deeply thankful to my family for their endless love, support, and understanding. They have been my pillars of strength, providing the emotional and moral support needed to persevere through difficult times. Their belief in my abilities and constant encouragement have been a driving force behind my efforts.

Navigating this thesis process has been demanding, but the combined support and encouragement from Dr. Soma Paul, my family, and friends have been essential in helping me achieve this milestone. Their unwavering belief in me provided the strength and resilience needed to see this project through to completion.

Lastly, I want to acknowledge the broader academic community at IIIT Hyderabad for providing an enriching environment that has fostered my intellectual growth. The resources, facilities, and academic discourse have significantly contributed to my research and overall experience.

This thesis stands as a testament to the collective effort of all the wonderful people who have been part of my journey. To each of you, I extend my deepest gratitude and appreciation. Thank you for your unwavering support and belief in my potential.

Abstract

This thesis explores three innovative projects aimed at advancing linguistic and semantic resource development in natural language processing (NLP).

The first project focuses on constructing a knowledge base tailored to user-oriented documents in the AC manual domain using the PurposeNet architecture. This initiative aims to systematically extract and organize domain-specific information from diverse textual sources, facilitating enhanced comprehension and retrieval of technical information crucial for user support and maintenance tasks.

The second project addresses the automated conversion of Indian language morphological processors into Grammatical Framework (GF). Grammatical framework (GF) is an open source software which supports semantic abstraction and linguistic generalization in terms of abstract syntax in a multilingual environment. This makes the software very suitable for automatic multilingual translation using abstract syntax which can be treated as a interlingua. As a first step towards building multi-Indian language translation system using GF platform, we aim to develop an automatic converter which will convert morphological processors available in various formats for Indian languages into GF format. As part of project, we develop a deterministic automatic converter that converts LTtoolbox and ILMT morphological processors into GF format.

The third project presents an Intelligent Interactive Editor designed to facilitate the creation of Controlled Natural Language (CNL) for Hindi text. CNL is a language-independent information system that captures accurate Syntactico-Semantic Representations of source languages, ensuring clarity and precision in language processing. The editor leverages multiple state-of-the-art tools and custom-built tools to auto-populate most of the fields required in the CNL format, significantly easing the user's workload. These tools are integrated using a plugin architecture, ensuring the editor remains future-proof and adaptable to new advancements. Additionally, the editor features a mapper layer that allows intermediate representations used by the tools to be mapped to the CNL format through various extendable rules, provided developers adhere to the required input-output format. CNL is a crucial foundational element for developing a robust multilingual natural language generation (NLG) tool that transcends the limitations of relying on a single source language.

Together, these projects represent significant advancements in linguistic and semantic resource development for NLP applications. They provide robust tools and methodologies that contribute to foundational research in semantic representation and computational linguistics, offering practical solutions to improve the performance and applicability of NLP systems across diverse linguistic contexts.

Contents

Chapter Page

List of Figures

Figure Page

List of Tables

Chapter 1

Introduction

The creation of linguistic and semantic resources in Natural Language Processing (NLP) plays a pivotal role in enhancing the accuracy and efficiency of language technologies. These resources form the bedrock that enables machines to comprehend, generate, and manipulate human language effectively. However, their development is accompanied by significant challenges, stemming from the intricate nature of linguistic structures and the diverse domains and languages they aim to represent.

Various methodologies—automated, semi-automated, and manual—are employed in their creation. Automated methods harness machine learning algorithms and computational techniques to extract patterns from vast datasets, offering scalability but requiring substantial computational resources and fine-tuning for accuracy. Semi-automated approaches combine automated tools with human intervention to ensure quality and address subtle linguistic distinctions that automated systems may overlook. Manual creation involves expert linguists and annotators who meticulously craft linguistic annotations, ensuring high precision but demanding greater time and financial investment.

These resources find application across diverse NLP domains such as machine translation, information retrieval, and sentiment analysis. In the context of Indian languages, characterized by vast linguistic diversity and underdeveloped computational tools, the creation of robust linguistic resources assumes heightened importance. These resources not only bolster the development of NLP applications tailored to Indian languages but also contribute to preserving and promoting linguistic diversity and cultural heritage.

As NLP advances alongside innovations in AI and machine learning, the demand for comprehensive linguistic resources continues to grow. By addressing the challenges and pain points in resource creation through innovative methodologies and interdisciplinary collaborations, researchers endeavor to narrow the gap between human language and machine processing capabilities. This introduction sets the stage for exploring diverse methodologies, their applications, and future directions in linguistic resource development, underscoring their pivotal role in advancing the frontiers of NLP and enabling sophisticated language technologies across diverse linguistic contexts. With this in mind, we will explore three such resources developed using various methodologies across different domains and the challenges encountered during their construction. Moving forward, we will divide this thesis into three sections, covering the development of three distinct resources - Chapter II: Building Knowledge base from User-Oriented Documents in PurposeNet Architecture in AC Manual Domain, Chapter III: Automatic conversion of Indian Language Morphological Processors into Grammatical Framework, Chapter IV: An Intelligent Editor to support creation of Controlled Natural Text for Hindi sentences and then we end with summary of how different methodologies, domain and user selection impacted the resource creation.

Chapter 2

Building Knowledge Base from User-Oriented Documents in PurposeNet Framework in AC Manual Domain

2.1 Introduction

User manuals of electronic appliances, widely available on the web, contain information related to their functionalities. Developers of user manuals are always concerned with making the display of the content user friendly using the interplay of images, graphics and necessary text. However, users would prefer a quick automatic retrieval of the necessary information, to the manual scanning of the whole document when the query is specific. When users want to know answers to their query as in a FAQ (frequently asked question), a Query Box facility on user manual page will definitely increase the usability of the resource. Retrieval of the relevant information is convenient when the knowledge is represented in a structured knowledge base in terms of concept-relation tuple in a hierarchical network.

This chapter proposes creation of knowledge base in PurposeNet [19] from user manuals and discusses challenges and issues involved in processing user manuals in order to semi-automatically develop the knowledge base. We will discuss the following in the following chapter with proper review of the relevant works:

- Architecture of PurposeNet.
- Content and varieties of representation of the information in user manuals.
- Issues raised while processing user manuals.

Kietz et al. (2000) [10] has done similar work which involved creation of a domain specific ontology, based on information from a corporate intranet. Shallow language understanding by creating automated FAQ answering has been created by Snieders (1999) [21]. Mayee et al. (2010) [11] have tried to automatically map textual information into PurposeNet. Issues concerning automation of artifact information have also been discussed by Singh and Srivastava (2011) [20]. Our work is different in terms of the amount of pre-processing required to create the ontology.

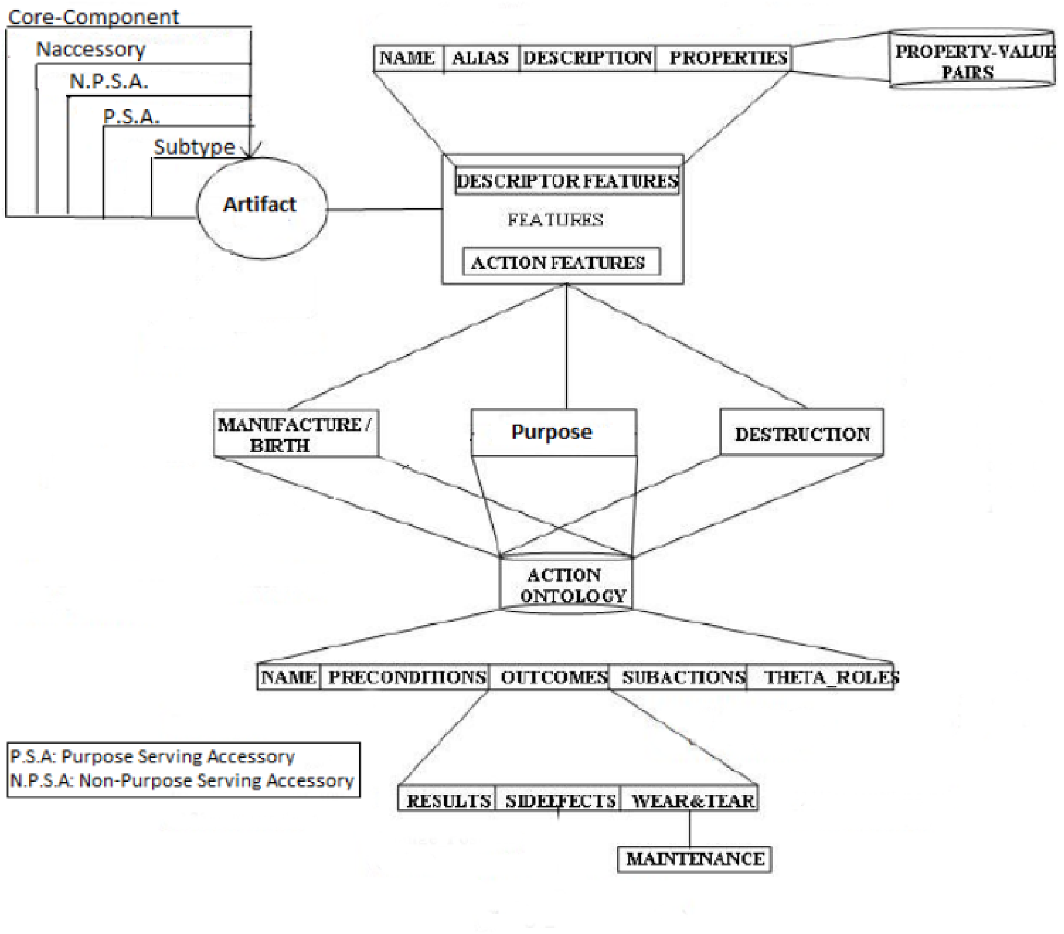


Figure 2.1: Structure of PurposeNet

2.2 Architecture

The ontology has been created using Protege which is an OWL ontology editor [12]. OWL consists of concepts, also known as entities, and relations. Concepts form a hierarchy with relations taxonomic in nature [11] with each concept having instances. Relations create non-taxonomic links between concepts or instances [13].

Architecture of the resource is based on the PurposeNet framework, the architecture of which is described in the trailing section. Its structure is described in figure 1.

2.2.1 PurposeNet

PurposeNet is a knowledge base of artifacts with purpose as the underlying design principle. Artifacts are fully described by its features and relationships with other artifacts.

2.2.1.1 Features: Descriptor and Actions

The various properties of an artifact are called its features. These features may be morphological, like the artifact's physical state, size, shape and magnitude. These are called descriptor features. Various activities associated with an artifact constitute its Action Features. This categorization has been developed based on the three major stages in the Life cycle of an artifact: Make/Birth, purpose-serving stage, i.e. Life, and Destruction. During the Life stage, the artifact may be in the general or repair-related maintenance stage. Every non-primitive action can be fully described using a quadruple consisting of its preconditions, outcomes, sub actions and semantic roles. We call this an action frame.

2.2.1.2 Relations

Different artifacts in the PurposeNet are related to each other using sub types, components, accessories and naccessories. The accessories are further divided into purpose serving and non purpose serving accessories.

Object	Type 1	Type 2	Type 3
Air conditioner	7	5	3
Freezer	11	2	2
Automobiles	9	2	4
Camcorder	6	6	3
Dryer	9	2	4

Table 2.1: This table shows classification of various user manuals based on the amount of information present from text, images, flowcharts, tables, etc. over five different entities with 15 manuals each

2.3 Types of User Manuals

User manuals are expected to be self-explanatory so that users can consult them and learn to use appliances. In order to achieve these goals, manuals often contain images and graphics. We have randomly selected 15 manuals of 5 artifacts[1] and identified these manuals into three types:

Type 1: Manuals with text as a major component of information.

Type 2: Manuals with boxes, tables and flowcharts of actions and artifact hierarchy.

Type 3: Manuals with images in between written text.

[1] Manuals have been selected from http://www.manualsonline.com/

Distribution of types for each of the five artifacts is presented in table 2.1. We note that mostly, the manuals are text based. Such observation motivated us to plan semi-automatic conversion of manuals into structured knowledge base.

2.4 Issues involved in Pre-processing stages

2.4.1 Presence of image in manuals

User manuals are meant to be descriptive and simple in nature. To achieve this, manuals often use images to depict important information like appliances, buttons and indicators. Automated conveying the meaning of these images in text form is difficult as images are not converted to text during the conversion. They get converted to garbage values which are of no use. Thus images are important in order to understand information regarding artifacts. Extracting such information is an issue with automation.

2.4.2 PDF to text conversion

Since text processing is necessary for interpreting the manuals, text conversion from PDF to a format compatible for text-processing is an inevitable task. As of now, there is no open source tool available which converts PDF to text with complete efficiency. Some information is lost. Also, after the conversion, spaces in between some words are deleted, thus creating merged words.

2.4.3 Domain specific properties

PurposeNet is an architecture which gives 25-26 basic properties to every artifact. But, further analysis on the user manuals revealed that every artifact shall have some domain specific properties, which may not be there in other artifacts. For AC, these properties are fan-speed, ac modes, etc. Also, every action involves preconditions and results which again are not actions but states. To resolve this issue, a new class of entities called states was introduced, which are related to concepts using object properties.

2.5 Issues involved in conversion

The next subsection examines the content of manuals of different models of AC in order to find out issues involved for automatic conversion of user manuals to PurposeNet architecture.

2.5.1 Categorization of content of manuals

Generalization of manuals is necessary for automation because the forms thus created can be used to create templates.

2.5.1.1 Nature of Content

It has been observed that user manuals typically convey information of the following categories.

- Morphological structure of artifact and its association with accessories
- Installation Guide
- Operation Guide
- Maintenance
- Safety Rules
- Troubleshooting (this can be in the form of FAQs or statements.)

13 manuals[2]of AC have been randomly selected, annotated by 3 annotators with the above categories and inter annotator agreement has been calculated for the two factors: Availability and Comprehensibility of information. This chapter will discuss in detail the criteria that has been used to calculate the two.

The availability was calculated by binary scoring for each manuals depicting the presence of data as 1 and nonavailability as 0. The comprehensibility was calculated by scoring on a scale from 0 to 4 where,

- 0 - Data not present, contributes 0% to the percentage score
- 1 - Present but incomprehensible, contributes 10% to the percentage score
- 2 - Less Information, contributes 60% to the percentage score
- 3 - Comprehensible but scattered, contributes 90% to the percentage score
- 4 - Completely comprehensible, contributes 100% to the percentage score

[2] Random selection was done from http://www.manualsonline.com/

Categories	Inf Avail	Chunk Avail	Comp. (%)	Comp. (Avg.Kappa Score)
Morphological Structure	0.92	0.85	85.90	0.429
Installation Guide	0.85	0.85	82.31	0.205
Operation Guide	0.85	0.85	73.33	0.374
Maintenance	0.85	0.85	75.13	0.412
Safety Instructions	0.92	0.77	86.15	0.169
Troubleshoot Instructions	0.85	0.85	83.85	0.685

Table 2.2: The table shows the availability and comprehensibility of different categories in the manuals.

Table 2.2 denotes average kappa score as well as average percentage score for each category taking all annotators into consideration[3] . The high availability and comprehensibility rate in Table 2.2 shows that a manual of the given domain can be chunked into these predefined categories.

2.5.1.2 Organization of Content

The order of the categories can vary from one manual to another. We notice that information can be organized in two ways:

- Action oriented: Instructions on the basis of what user wants to do[4]
  1. Setting Time for the First Time
     (a) Insert the batteries.
     (b) Push the battery cover on the rear of the remote control with your thumb and remove it.
     (c) Press the time button.
     (d) Use up and down arrows to set time.
     (e) Press the time button.
     (f) Result: The clock starts.
- Or Artifact oriented: Classifying buttons and specifying what each of them do[5]
  1. Remote controller
     (a) ON/OFF button :used to start and stop
     (b) TIMER button :used to set timer

[3] Annotation was done by 3 annotators all having fairly good knowledge of PurposeNet and automation tasks

[4] http://downloadcenter.samsung.com/content/UM/200305 /20030524161419000 ASM070VE.pdf

[5] http://www.palsonic.com.au/customer service/aircon instruct /Airmaster AOS Series Split Aircon Manual.pdf

(c) UP button :used to increase temperature and set time

(d) DOWN button :used to decrease temperature and set time

Or some level of combination of these two. Algorithm 1 describes how information from different types of manuals can be extracted and populated in PurposeNet. A schema of the ontology thus created is given in figure 2.2.

Algorithm 1 generateOntology(manual)

```
Identify the orientation of manual (artifact, action, other)
if orientation is action then
    Create the action ontology, using the given actions
    Identify its relations with the other actions and states
    Identify the thematic roles of those actions and states
    Execute steps 3.1 and 3.2
else if orientation is artifact then
    Create the artifact ontology, using the given artifacts
    Identify its relation with other artifacts
    Identify the purpose, birth, maintenance, installation, and destruction
    Execute steps 2.1 and 2.3
end if
```

2.5.1.3 Soundness and completeness of the knowledge base

In order to achieve completeness in PurposeNet, all information must explicitly be stated. This is achieved by verifying that all relations have been specified and also by using disjoint property between all the classes. For example, the manual does not mention the pre-condition that AC is on for the following actions:

1. Set swing mode on

2. Change fan speed

In PurposeNet, the pre-condition relation will be specified for the above two actions. Soundness is achieved in the ontology by specifying appropriate properties for the relations in PurposeNet. For example,

1. Inverse relation

    - is subaction of and has subaction

2. Transitive relation

    - has core component

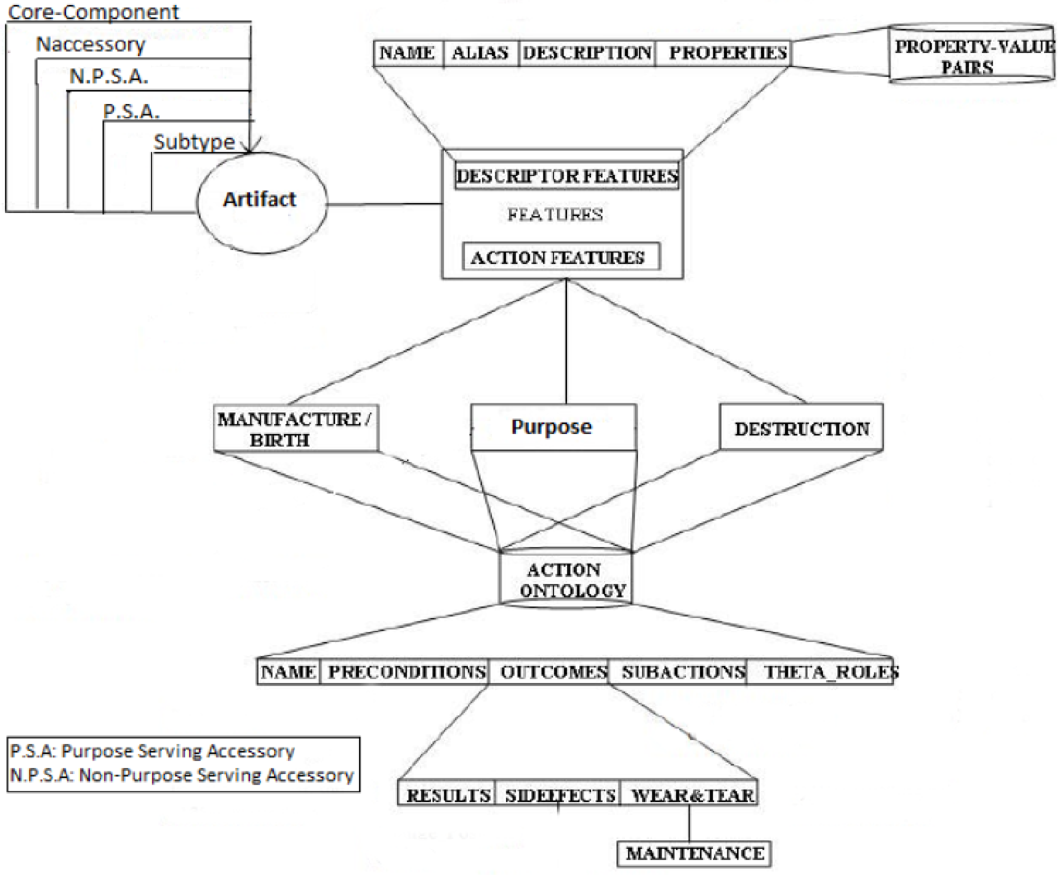


Figure 2.2: Text processing on user manuals having different ways of depicting information, and merging into one ontology.

3. Non-transitive and non-symmetric relation
    - has subaction

2.6 Challenges

Conversion of manuals to ontology semi-automatically involves the following challenges:

2.6.1 Unambiguous representation of Knowledge

Users manuals have two kinds of ambiguities. First, the same information can occur in different ways, introducing redundancy. For example,

- In the Samsung AC manual, temperature change and fan speed control is depicted in all the modes even when the process is the same.

Second, manuals can use different synonymous words to express an idea in the same or different documents. For example,

- Starting an AC has been written in two ways in manuals

  1. set AC on
  2. start AC

A language processing tool that will measure semantic similarities of concepts is required while converting the user manual into a knowledge base.

2.6.2 Difference in functionalities of different Artifacts

Present day air conditioners have more functionalities than the older models owing to evolution of the model and the company's selling points. For example, a Samsung model has an added 'turbo' cooling function that sets the AC in cool mode for a small time after which it brings AC back to its previously selected mode. Such Unique Functions Require manual intervention in order to understand and add them in the ontology.

2.6.3 Deciding the depth of Ontology

Deciding the depth in ontology is a serious concern. For example,

- For the action Change AC mode to Cool, the subaction can go to the depth

  1. press X button or
  2. compressor throw air

For such cases, the decision is taken based on the usability factor from the user's perspective. When a user tries to understand how to perform Change mode, for him/her the necessary information is to press the button and not the mechanical details of the action.

2.7 Conclusion

Our investigation revealed significant challenges in the pre-processing stages, PDF extraction, which highlighted the need for more advanced tools to ensure comprehensive ontology development. Addressing these challenges, we introduced domain-specific properties like fan speed and AC modes, enriching the ontology's specificity and utility for AC systems.

Furthermore, our approach to categorizing and organizing manual content underscored the importance of standardized frameworks for extracting and integrating diverse manual formats into a cohesive ontology structure. Through rigorous categorization and annotation processes, we validated the completeness and accuracy of the generated ontology.

Looking ahead, future research should focus on refining semantic similarity tools for concept disambiguation, and customizing ontology depth to meet user-centric usability requirements. These advancements are crucial for extending PurposeNet's application across broader technical domains, thereby advancing knowledge management and usability in automated systems.

In conclusion, this study demonstrates the effectiveness of PurposeNet in automating ontology creation from technical manuals, paving the way for enhanced knowledge representation and management in the realm of technical documentation and beyond.

Chapter 3

Automatic Conversion of Indian Language Morphological Processors into Grammatical Framework

3.1 Introduction

Many NLP resources have been developed for processing Indian languages in the last decade. A major consortium of Indian language Indian language MT system (ILMT) has been successfully carried out for 9 language pairs. These systems mainly follow a transfer based approach. The MT system from English to various Indian languages have also been developed through large projects. Morphological analyzers and generators have been developed as part of these projects as well as independently for many Indian languages. In this chapter, we present an automatic converter that converts morphological processors that exist in different formats into one common format of Grammatical Framework.

Grammatical framework (GF) is an open source software which supports semantic abstraction and linguistic generalization in terms of abstract syntax in a multilingual environment [18]. Abstract syntax can be viewed as an interlingua in the context of multi-lingual machine translation. Apart from abstract syntax there exists another module called concrete syntax in which the morphological specification and syntactic behavior of a language can be captured. The abstract syntax and concrete syntax of all languages that one wants to handle together can produce very good quality, especially for domain based MT systems. This is the motivation for converting morphological processors, that at present exist in different formats for different Indian languages, into GF format so that good quality meaning driven multi-lingual MT can be accomplished.

Our morphological converter presently converts morph resources designed in LTtoolbox and ILMT morph framework into GF format. We have experimented with Hindi, Oriya, Tamil, Marathi and Sanskrit morphological processors. We have achieved 100% information preservation for the conversion of Hindi, Tamil and Oriya languages. Sanskrit and Marathi LTtoolbox resources are quite huge and we have encountered some issues in conversion which we will discuss in this chapter.

The chapter is divided into the following sections. In the next section, we will briefly introduce LTtoolbox, the structure of morph used in ILMT and functionalities required for representing morphological information in GF. Section 3 presents related work. Section 4 presents our approach of automatic conversion. In section 5, we will present the result of automatic conversion and analyze the results for Hindi, Oriya [9] and Sanskrit. We conclude the chapter by presenting a critical review of our work and also insight into future work.

3.2 Frameworks

We will briefly discuss the architecture of the morphological processors of the three frameworks, Grammatical Framework, Lttoolbox and ILMT.

3.2.1 Grammatical Framework

GF is a development platform which allows linguists to build grammars [16] [17]. It uses a paradigm approach in its morph resource. Parameters and Operations are used as building blocks to create morph resources. A parameter is a user-defined type which is used to model lexical features like number, gender etc.

```
param Number = Sg | Pl
```

Listing 3.1: Example Parameter

Here, Number is a parameter representing the singularity/plurality of given word. Each parameter has a set of constructors, which represent possible values of that parameter. In this case, Number has two constructors namely Sg and Pl, which are declared as shown in the example above. An Operation is a function which takes a lemma of the given word and generates a table consisting of all possible word forms. A table in operation consists of branches with constructors on the left and corresponding word forms on the right. Table is computed by pattern matching which returns the value from the first branch whose pattern matches the argument [15].

```
oper regNoun : Str → {s : Number ⇒ Str
    } = \dog → {s = table {
        Sg ⇒ dog ;
Pl ⇒ dog + ”s”
}};
```

Listing 3.2: Example Operation

In the example shown above, the operation regNoun takes a String as an argument and returns a Number to String s: Number =¿ Str . If the string dog is passed to the operation then it returns a paradigm stating that when the word is singular(Sg) , the operation returns the same word(i.e. dog), if it is plural(Pl) then it returns dogs.

3.2.2 LTToolbox

Lttoolbox is an open source finite state toolkit used for lexical processing, morphological analysis and generation of words [6]. Like GF, Lttoolbox also uses the paradigm approach for creating morphological analyzers . Morph resource in Lttoolbox has three important sections (i.e. for automatic conversion) namely symbol definition section, paradigm definition section and lexicon dictionary. Symbols are used to define lexical categories, features and their values in the morph. These symbols are defined within symbol definition (sdef).

```
⟨sdef n=”gen:m” c=”masculine” /⟩
```

Listing 3.3: Example Symbol definition

In the above example, gen is the symbol for gender, m is the corresponding feature value. Paradigm takes a lemma and generates all possible word forms. Each paradigm consists of several entries. Each entry has the data to create a word form for a given set of grammatical symbols. These paradigms of the grammar are defined within the paradigm definition (pardef).

```
⟨pardef n=”kAl/A adj”⟩
⟨e⟩⟨p⟩⟨l⟩e⟨/l⟩⟨r⟩A⟨s n=”cat:adj”/⟩⟨s n=”
    case:o”/⟩⟨s n=”gen:m”/⟩⟨s n=”num:s
    ”/⟩⟨/r⟩⟨/p⟩⟨/e⟩
⟨/pardef⟩
```

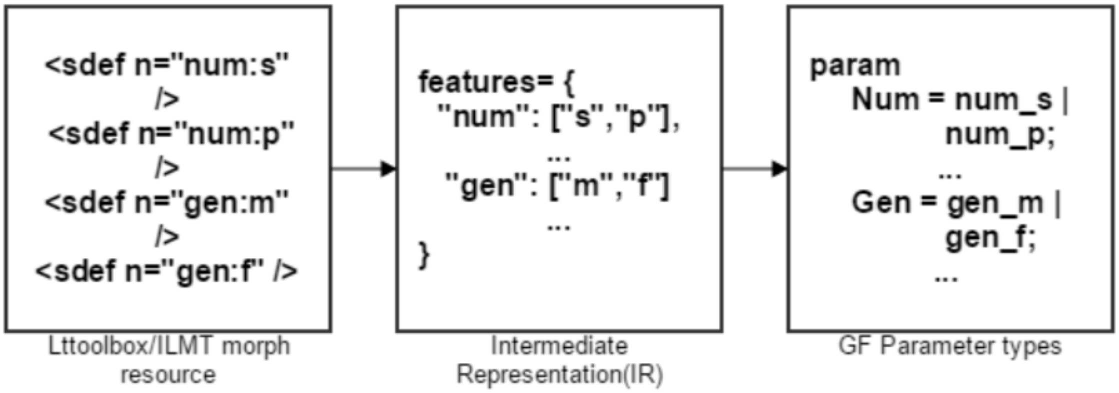
Listing 3.4: Example Paradigm Definition

In the above example, the lemma in consideration is kAlA. For the given entry (e) , the word form kAle is generated for kAlA when the grammatical symbols are

```
⟨cat=adj;case=o;gen=m;num=s⟩
```

3.2.3 ILMT

ILMT uses Computational Paninian Grammar (CPG) for analyzing language and combines it with machine learning. It is developed using both traditional rules-based and dictionary-based algorithms with statistical machine learning [22]. ILMT morph resource has categories, features, feature values and word forms. Each category with its name and its corresponding features are defined in a single file.



```
<sdef n="num:s"
/>
<sdef n="num:p"
/>
<sdef n="gen:m"
/>
<sdef n="gen:f" />
```

```
features= {
  "num": ["s","p"],
          ...
  "gen": ["m","f"]
          ...
}
```

```
param
   Num = num_s |
          num_p;
          ...
   Gen = gen_m |
          gen_f;
          ...
```

Lttoolbox/ILMT morph resource
Intermediate Representation(IR)
GF Parameter types

Figure 3.1: Overview of the data transformation

Similarly, each feature along with its name and value are stored in a different file. Each category has its own paradigm file. A category can contain many words. Each word belonging to a category is stored in its paradigm file with its root and all its forms. These word forms are listed in last varies first order. For example,

- Noun_m g_m case num means that Noun m is a category depending on features g_m, case and num.
- case d o represents that case feature has values d and o. Similarly, g_m has value m. num has values s and p.
- Noun m Gara, Gara, Gara, Gara, GaroM/Garoz . means that the first word is the root form, and the rest are its word forms in the last varying fast order of category features

  ```
  (i.e, ⟨m,d,s⟩, ⟨m,d,p⟩, ⟨m,o,s⟩, ⟨m,o,p⟩).
  ```

3.3 Related Work

There's little work done on automatic resource sharing between frameworks. Some notable works include resource sharing between Apertium and Grammatical Framework [5]. In this chapter, we propose an automatic approach to extract Apertium shallow transfer rules from a GF bilingual grammar. The process of creating GF data from Lttoolbox grammar is done manually. They successfully created and tested the system with an English-Spanish language pair. Also some work has been done to import Indian languages [8] into GF. However, this is a manual process and it requires linguists. To the best of our knowledge, there exists no work addressing the automatic conversion from existing morph resources(ILMT/Lttoolbox) of Indian languages into GF.

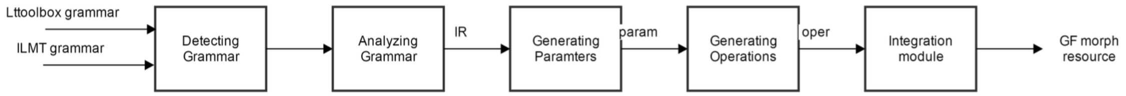


Figure 3.2: Modular level overview of our Approach

3.4 Our Approach

In this section, we describe our approach of converting LTtoolbox and ILMT morph resources into GF morph resources. The steps of conversion is presented in the figure 3.2. Our approach converts the morphological processors available either in XML or ILMT syntax format and converts them into Grammatical Framework. The key idea here is that Grammatical Framework is a programming language with a definite syntax and we need to transform the given morphological processors such that they adhere to Grammatical Framework's syntactic structure. This calls for the need to change morphological processors into non-ambiguous, permissible units which are supported by Grammatical Framework. Some examples of above mentioned challenges are as follows. Further details regarding these decisions are described in algorithms mentioned below.

As already mentioned in above sections, Lttoolbox supports paradigms which depend on variable number of parameters whereas Grammatical Framework only supports dependence on pre-determined parameters. So we need to analyze all paradigms and their dependencies to find the specific occurrences and transform them into tabular syntax structure, supported by Grammatical Framework. Some attributes present in the source files (e.g. numeric attributes, duplicate attributes) cannot be transformed directly into GF e.g. parsarg = 0. We chose to modify the parameters to include the actual parameter value and other contextual information e.g. the parameter type, the paradigm name etc.

3.4.1 Detecting Grammar

In this step, we classify the user input as Lttoolbox or ILMT morph resource. This is a relatively simple step because of widely different syntax of both morph resources. We accomplish this using syntax-based heuristics.

3.4.2 Analyzing Grammar

In this step, we take the respective grammar, parse it and then convert into an intermediate representation (IR). The motivation behind this step is to reduce complexity for the following steps i.e Generating parameters and operations. In the case of ILMT, the morph resource is present across different sources. So we convert it into XML using the same structure as in Lttoolbox, to make the conversion process more generalized. Once we have the XML source, we will convert it into IR by using the algorithm shown in Algorithm.2.

Algorithm 2 Algorithm for Analyze Grammar

```
features =                                          ▷ sdef represents the set of all symbol definitions
for all s ∈ sdef do                                 ▷ s contains lexical features and lexical values
    features[s.lexical-feature].add(s.lexicalvalue)
end for
paradigms =                                         ▷ pardef represents the set of paradigm definitions
paradigms[root].add(e.featureset, e.word-form)
return features, paradigms
```

In algorithm 2, we are converting symbol definitions and paradigm definitions into IR (i.e features and paradigms).

3.4.3 Generating Parameters

In this step, we use the features from IR to generate parameters and their constructors, using the algorithm shown in Algorithm 3. The function buildParameters generates GF syntax (parameters and corresponding constructors) for a given feature name and its values. For example calling the function buildParameters with arguments (Num,[num s,num p]) returns

```
Num = nums | nump;
```

Listing 3.5: Example Parameter Syntax in GF

3.4.4 Generating Operations

In this step, we use the IR (features, paradigms) to generate operations using the algorithm given in Algorithm 3. The function buildOperations builds operations for each paradigm. As explained in the above section, each operation consists of declaration and description which are generated by buildDeclaration and buildTable respectively. To do this we need to keep track of all the features used for a paradigm. So we built a helper function which does this by iterating through all the entries in the paradigm.

For example, calling buildDeclaration with arguments (case,gen,num) produces the

```
    x1,,,,,,x8:Str → Case ⇒ Gen ⇒ Num ⇒ Str
```

(we used x1-x8 because each feature has two values, total possible values in table are 2x2x2 = 8). The function buildTable recursively builds the table which is used to generate the possible word forms for a given lemma in GF, using the algorithm shown in Algorithm 4.

Algorithm 3 Algorithm for generating operations

```
function BUILDOPERATIONS(features, paradigms)
    for all p ∈ paradigms do
        buildDeclaration(p, dependentFeatures)
        buildTable(features, p, dependentFeatures)
    end for
    return
end function
```

Algorithm 4 Algorithm for building table

```
function BUILDTABLE(features, paradigm, dependentFeatures)
    if dependentFeatures is empty then
        return
    end if
    for all df ∈ dependentFeatures do
        for all v ∈ features[df] do
            v ⇒ buildTable(features, paradigm, rest(dependentFeatures))        ▷ rest(1,2,3) = 2,3
        end for
    end for
end function
```

3.4.5 Code and Datasets

The entire code base has been written in Python. The description of datasets are shown in Table 3.1. Code and datasets are publicly available under open-source license [1].

3.5 Evaluation and Error Analysis

We measure the accuracy of the converter in the following way. The same text is given to both GF analyzer and the original source tool (LTtoolbox and ILMT). We verify that the output of GF analyzer and the original source tool is identical. Thus we ensure that information is faithfully transferred. We found that there is no loss of data for the chosen languages i.e Hindi, Oriya, Tamil and Bengali. This evaluation is repeated for various random news articles and 100% information was preserved in the output. However, we have come across some problems in converting Sanskrit and Marathi morph resources. These problems are mainly caused due to the presence of language specific syntax which we had trouble in generalizing. Even though we had some troubles, we have found that, after analyzing files manually, our approach will still be able to make the automated conversion to around 65%.

[1] https://github.com/harshavardhangsv/automaticMorphResourceConverter

Dataset	# of paradigms	# of words
Hindi	57	26356
Tamil	254	34290
Oriya	45	2860
Bengali	148	5478

Table 3.1: Number of paradigms and words converted in each dataset

3.6 Conclusion and Future Work

In this chapter, we have explained the process of creating an automatic converter which deterministically converts morphological processors from different formats into GF format without any information loss for Hindi, Oriya and Tamil. We are presently working on other Indian languages such as Marathi, Sanskrit and Telugu. We have observed that if there exists any language specific morpho-syntactic information in the resource, conversion of such resources requires additional work. At present we are converting morph resources created in Lttoolbox and ILMT morph format. In the future we might come across a morphological processor developed in another framework. In order to bring the complete genericness into our tool, we want to modularize our system. We intend to develop a generic system that first asks the user about the format of their morphological processor and then activates the right module for the conversion. We hope that with our effort we will be able to offer a NLP resource to the community which will eradicate the barrier of framework differences.

Chapter 4

An Intelligent Interactive Editor to Support Creation of Controlled Natural Text for Hindi Text

4.1 Introduction

The growing complexity and diversity of natural languages present significant challenges in the fields of natural language processing (NLP), machine translation, and information retrieval. Accurate syntactico-semantic representation of source languages is crucial for these applications to function effectively. Controlled Natural Language (CNL) systems have emerged as a promising solution, providing a structured and precise way to represent linguistic information while maintaining human readability. However, creating and maintaining CNL can be labor-intensive and requires considerable expertise.

This chapter introduces an Intelligent Interactive Editor specifically designed to support the creation of CNL for Hindi. The editor aims to simplify the process of generating CNL by leveraging a combination of state-of-the-art tools and custom-built solutions. By automating the population of most required fields, the editor significantly reduces the manual effort involved in creating CNL. Its plugin architecture ensures that the system remains adaptable and future-proof, allowing for seamless integration of new tools and technologies as they become available.

A key feature of the editor is its mapper layer, which enables the transformation of intermediate representations used by various tools into the CNL format. This is achieved through a set of flexible and extendable rules, allowing developers to enhance the system by following a defined input-output format. The result is a robust and scalable tool that not only facilitates the creation of CNL but also ensures that the generated representations are accurate and consistent.

The remainder of this chapter is organized as follows: Section 4.2 reviews related work in the field of CNL and existing tools for linguistic representation. Section 4.3 provides an overview of the Intelligent Interactive Editor, detailing its architecture, components, and functionalities. Section 4.4 discusses the implementation and integration of the state-of-the-art and custom-built tools. Section 4.5 and 4.6 discuss the internal tools created to support the interface. Section 4.7 concludes the chapter with evaluation and outlines potential future work.

By providing a comprehensive tool for the creation of CNL, this work aims to bridge the gap between natural language and machine-readable formats, thereby enhancing the performance and applicability of NLP systems in processing Hindi text.

4.2 Related Work

The development of Controlled Natural Language (CNL) tools represents a significant advancement in computational linguistics and Natural Language Processing (NLP), aimed at enhancing the clarity and precision of technical communication. Tools like WebLicht [23] and Attempto Controlled English (ACE) [7] have been instrumental in defining methodologies for CNL generation. WebLicht provides a comprehensive platform for linguistic data processing, facilitating syntactic and semantic analysis crucial for generating structured CNL texts. ACE, on the other hand, stands out for its role in establishing standards for controlled language specifications, enabling automated reasoning and text analysis by ensuring texts are both machine-understandable and human-readable.

Similar efforts have been made for Indian languages with tools like Transzaar [1] and Katha [14]. Transzaar is a computer-aided translation tool that supports multiple Indian languages, facilitating translation between pairs such as English-Hindi, Hindi-Urdu, and Hindi-Punjabi. It aims to empower human translators by providing outputs from multiple machine translation systems, thus enhancing the translation quality through human oversight. Katha, on the other hand, is a visual programming interface designed to simplify the creation of linguistic resources and interactive applications for Indian languages. While both tools contribute significantly to the field, they differ in purpose compared to our interactive editor, which focuses on the creation and maintenance of CNL for Hindi text

In the realm of semantic representations, frameworks such as Abstract Meaning Representation (AMR) [2] and Minimal Recursion Semantics (MRS) [4] have extended the capabilities of CNLs by capturing deeper semantic structures. AMR offers a graph-based representation that abstracts away from syntactic variation, focusing on capturing core semantic content across languages. MRS, developed under the framework of Minimal Recursion Semantics, provides a formalism for semantic representation that integrates syntactic and semantic information, supporting precise analysis and generation of natural language.

Evaluation methodologies for CNL generation tools are pivotal in assessing the quality and effectiveness of generated outputs. Benchmarks like the Semantic Textual Similarity (STS) dataset [3] offer standardized assessments across various domains and languages, providing insights into the robustness and applicability of CNL generation tools in different contexts.

4.3 CNL Text Processing

To create an accurate Syntactico-Semantic Representation of Hindi text, our Intelligent Interactive Editor supports the construction of Controlled Natural Language (CNL) representations. The CNL approach ensures clear and precise documentation of linguistic elements and relationships. Below, we elaborate on the core components and the user interface structure designed to facilitate this process.

1. **Concept Representation**
   Concepts in a sentence are distinctly represented to capture details such as postpositions, Tense-Aspect-Modality (TAM) information, word repetition, adjectives, and compound nouns.

   Example: eka Cota baccA is represented as eka CotA baccA_0.

2. **Concept Indexing**
   Concepts are indexed unambiguously to identify their sense in the context of the sentence.

   Example: baccA is indexed as baccA_1.

3. **Column Indexing**
   A straightforward indexing system is applied to the possible columns serially.

4. **POS Identification in CNL**
   The CNL framework identifies parts of speech (POS) such as proper nouns (propn), mass nouns (massn), definite nouns (defn) and various other categories.

5. **Gender-Number-Person (GNP) Information**
   GNP details are included for nouns to provide context about gender, number, and person.

   Example:[m sg 3] indicates male, singular, third person.

6. **Concept Relations**
   Relationships between nouns and their modifiers, and verbs and their modifiers, are denoted in a single concept.

   Example: eka CotA baccA_0 becomes 1.3:saMKyA-viSeRaNa 1.3:viSeRaNa.

7. **Karaka Relations**
   The relations between events and their participants are identified, such as K1 representing the agent (karta).

8. **Sentence Type**
   Sentences are classified as either affirmative or interrogative.

4.4 Interface Design

The user interface (UI) of our editor is structured into multiple sections to streamline the user interaction, displaying one section at a time to reduce cognitive load.

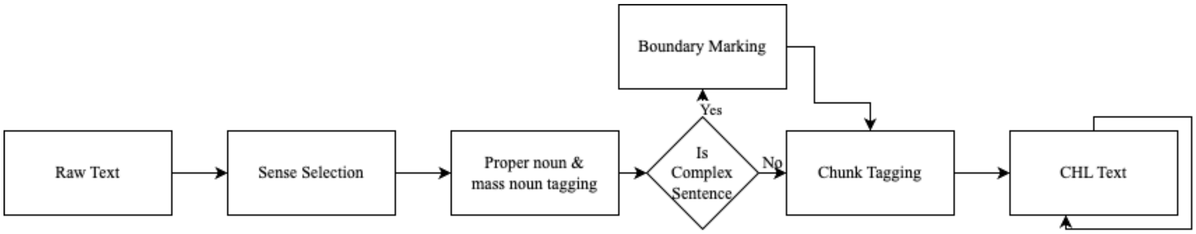


Figure 4.1: Flow chart of user actions in Communicate

4.4.1 Section 1: Input Section

Users can input Hindi text through various methods, including transliteration or direct Hindi scripting (Lipyantaran, Bolnagiri, Phonetic, Inscript). Proper nouns and boundary markings are identified using POS tagging.

4.4.2 Section 2: Mark Proper, Definite, and Mass Nouns

Users mark coarse tags for nouns, which helps in disambiguating and providing essential information for natural language generation.

User Actions:

1. Select all the nouns that relate to a single noun type
2. Click on the noun type that needs to tagged to the selected nouns

Automated Assistance: POS tagging pre-identifies nouns to facilitate user annotation.

4.4.3 Section 3: Sense Selection

Disambiguation of words with multiple meanings is essential. Users select the correct sense based on a pre-compiled concept gloss dictionary.

Dictionary Creation: Utilizes an English to Hindi linked synset dictionary.

4.4.4 Section 4: Mark Boundaries of Simple Sentences

To accurately identify boundaries of simple sentences within complex sentences, we utilize outputs from two different parsers: ILMT Parser and Anusaaraka's rule-based parser.

- **ILMT Parser**: If the root word is tagged as CCP tag and it has two children connected by CCOF dependency relation , the sentence is marked as complex. To find the boundaries of the simple sentences, we reconstruct the dependency trees of both children and identify the leaf nodes of the extremities.

- **Anusaaraka's Rule-Based Parser**: This parser outputs clips facts, where keys represent conjunction-components and values are lists of multiple conjunction word indices. By parsing these facts, we identify the boundaries of simple sentences by considering the previous indices as the end of the sentence.

User Actions:

1. Confirm: Approves the suggested boundaries.
2. Modify: Allows manual adjustment of boundaries.

4.4.5 Section 5: Mark Chunks and CNL Concept Creation

To accurately mark chunks, the process utilizes the ILMT (Indian Language Machine Translation) chunker tagger output, parser output, and a custom mapper. The custom mapper plays a crucial role in transforming the representation of modifiers to the chunk root word into a CNL concept representation. The transformation rules and symbols used include:

4.4.6 Section 6: CNL Text Editing Section

This section allows users to directly edit any row they find incorrectly populated, ensuring that the CNL (Controlled Natural Language) text is accurate and reliable. The following describes how previous sections contribute to this editing process and the specific actions users can perform.

1. **Concept Row Creation**
   Concept rows are generated based on information provided in the tag chunks section. Each chunk represents a meaningful unit or concept extracted from the text. This ensures that all significant elements are accurately represented in the CNL.

2. **Concept Index Row Creation**
   The Concept Index row is populated using the Sense Selection Section. This section determines the specific sense or meaning of a word within its context, ensuring that the correct interpretation is chosen and indexed appropriately.

3. **Column Index Creation**
   Column indices are created according to the information provided in the tag chunks section. The number of chunks is counted, and indices are assigned serially. This systematic indexing helps maintain the structure and order of the concepts.

4. **POS Tagging**
   Part-of-Speech (POS) tagging is performed using information from the Mark Proper Noun tagging section. Proper nouns are tagged explicitly, while other concepts are tagged using finer tags

provided by the POS tagger. Additionally, place and location nouns are tagged separately using a lookup into an in-house database created by Anusaaraka Lab linguists. This database includes location and time information, ensuring precise tagging.

5. **Gender Number Person Info**

   This row is auto-populated using the output from the ILMT (Indian Language Machine Translation) pipeline. The morph output provides information about gender, number, and person (GNP), which is crucial for accurate linguistic representation.

6. **Concept Relations**

   Concept relations are tagged by mapping chunker and parser information to the appropriate concept relations defined in CNL. For example, an adjective plus a number is mapped to ”saMKyA-viSeRaNa” (numerical adjective). This ensures that relationships between concepts are clearly defined.

7. **Karaka Relations**

   Karaka relations, which denote syntactic and semantic roles of nouns in relation to verbs, are tagged by mapping rule-based parser output to the corresponding relations in CNL. Editable fields in this row provide intelligently sorted karaka relations, showing a descending order of possible relations based on their likelihood.

   Update Action: For editable fields in the Karaka Relations row, users can select from intelligently sorted karaka relations. This sorting is based on the likelihood of each relation, making it easier for users to choose the most appropriate one.

   Creating Weighted Karaka Relations:

   (a) **Initial Analysis**:

   By analyzing the Sampark parsed corpus, a dictionary of pratyaya (suffixes) and their corresponding karaka relations is created. This dictionary includes the count of each relation a pratyaya is attached to a verb

   (b) **Learning and Updating:**

   The system learns and updates weights for pratyaya-karaka relations based on user interactions. Every new text created using the editor and every correction provided by the user are used to update and improve the weight mappings, enhancing accuracy over time.

4.4.7 Section 7: Feedback Section

This section is designed to capture user feedback and facilitate the continuous improvement of the CNL text generation tool. It allows users to report issues and suggest improvements, which are then analyzed and used to refine the system. Here are the detailed functionalities:

1. **User Input Field**

- **Purpose**: This input field allows users to provide manual feedback on what they perceive as incorrect or problematic with the interface actions
- **Integration with Logs**: Each feedback entry is attached to the input texts and the log journal of all actions and tool outputs. This comprehensive logging ensures that developers can easily trace and debug the issues identified by users.
- **Example Use Cases**
    - A user notices that a specific tag was incorrectly assigned and provides feedback explaining the error.
    - A user experiences an unexpected interface behavior and describes the scenario in detail.

2. **Manual Changes and Data Aggregation**

- **Tracking Manual Changes**:
    - Every manual change made by the user in the CNL Text section and the chunk section is tracked.
    - These changes include corrections to concept rows, POS tags, concept relations, karaka relations, and any other editable fields.
- **Data Aggregation**:
    - The information from these manual changes is aggregated into bulk data sets.
    - These data sets are organized in a structured format, capturing the nature of the change, the original output, and the corrected output provided by the user.
- **Sending to Tool Maintainers**:
    - The aggregated data is sent to the respective tool maintainers or developers.
    - This data is invaluable for maintainers as it highlights areas where the tool's performance can be improved and provides real-world examples of the tool's usage and its shortcomings.

4.5 Mapper

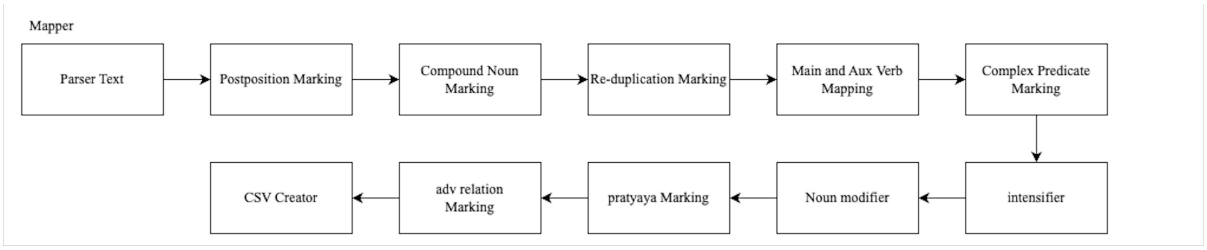


Figure 4.3: Overview of Mapper

The Mapper is a crucial component designed to preserve the syntactic and semantic structure of the original sentence while transforming it into the CNL format. Here's how it works:

4.5.1 Representing CNL Format While Retaining Original Syntax

The Mapper ensures that the original syntactic relationships and dependencies are retained when converting sentences into CNL format. This involves maintaining the correct word order, hierarchical structures, and dependencies as they appear in the source text.

4.5.2 Marking Intra-Concept Relations

Using a set of transformation rules, the Mapper accurately marks relationships within concepts in the CNL format. These rules define how various syntactic or semantic features, such as adjectives modifying nouns, are represented in CNL. By applying these rules, the Mapper ensures that intra-concept relationships are clear and consistent.

4.5.3 Representing All Rows in CNL Format

The Mapper processes each row of the CNL representation, ensuring that all elements, such as POS tags, dependency relations, and concept relations, are correctly formatted according to CNL standards. This comprehensive approach ensures that every aspect of the sentence is accurately captured in the CNL format.

4.5.4 Providing Mapped POS Tags and Relations

The Mapper maintains a database that maps all possible parser relations to their corresponding CNL relations. When processing a sentence, the Mapper refers to this database to ensure that each POS tag, dependency relation, and concept relation is correctly identified and represented in the CNL format.

4.5.5 Transforming Between Formats

The Mapper facilitates the conversion of CNL format into the input format required by various tools and vice versa. This bidirectional transformation capability ensures that data can be seamlessly passed between the CNL system and the external tools used for processing.

4.6 Tool Wrapper

The Tool Wrapper serves as an intermediary, managing the interaction between user inputs, the Mapper, and various tools. It plays several key roles:

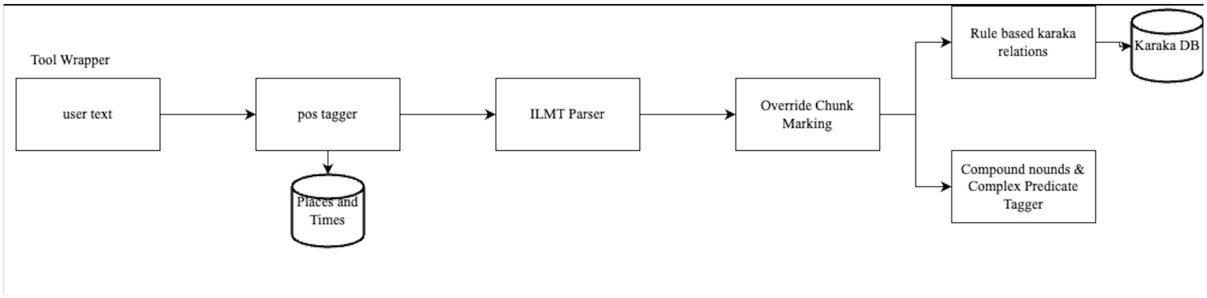


Figure 4.4: Overview of Wrapper

4.6.1 Invoking the Tool Wrapper at Each Step

The Tool Wrapper is invoked at each stage of the CNL text generation process. This ensures that all necessary transformations and tool executions are managed correctly and that the workflow proceeds smoothly from one step to the next.

4.6.2 Handling User-Provided Information

The Mapper collects all information provided by the user, such as edits and annotations, and passes this to the Tool Wrapper. The Tool Wrapper then transforms this information into the appropriate format required by the various tools. This ensures that user inputs are accurately processed by the tools.

4.6.3 Managing Custom Tools

In addition to standard external tools, the Tool Wrapper also handles the execution of custom-developed tools. These tools, which might be specifically designed for tasks such as proper noun identification in Hindi, are integrated into the system by the Tool Wrapper. It ensures that these tools are correctly invoked with the appropriate inputs and that their outputs are incorporated back into the system.

4.6.4 Integrated Workflow

The combined operation of the Mapper and Tool Wrapper ensures a seamless and accurate CNL text generation process. The Mapper preserves syntactic and semantic integrity while transforming data, and the Tool Wrapper manages tool interactions and data transformations. Together, they ensure that user inputs are accurately processed, tools are correctly utilized, and outputs are integrated into the CNL format, resulting in a robust and efficient workflow.

4.7 Evaluation and Future Work

4.7.1 Evaluation

4.7.1.1 CNL Text Creation

More than 150 CNL texts were created using the interface. This extensive usage provided a significant dataset to evaluate the performance and effectiveness of the system.

4.7.1.2 Evaluation Criteria

The evaluation criteria were established to measure the accuracy and effectiveness of the editor. The primary criterion was whether the final CNL text required any changes from the initially generated CNL text by the editor. If no changes were necessary, it indicated that the intelligence of the editor performed well. Any deviation or correction identified an error in the editor's output.

Each type of error was assigned a weight, and the overall error rate was calculated. According to this evaluation, the system exhibited about a 30% error rate. This means that 30% of the generated CNL texts required manual corrections, highlighting areas where the editor's algorithms and heuristics need improvement.

4.7.1.3 Architecture and Design Strengths

Despite the error rate, the architecture and design decisions facilitated easy identification and analysis of errors. The system's structure allowed linguists and tool maintainers to quickly pinpoint and understand the nature of errors. This capability is crucial for iterative improvement and refining the system's performance.

4.7.2 Code and Datasets

The entire code base has been written in Python, Javascript, HTML, CSS, databases - sqlite3, mysql, . The description of datasets are shown in Table 3.1. Code and datasets are publicly available under open-source license [1].

4.7.3 Challenges Identified

4.7.3.1 Complex Sentence Formation

One of the main challenges identified was the difficulty in correctly tagging information for complex sentences. The existing tools struggled with accurately processing these sentences, necessitating manual intervention. This highlights the limitations of the current tools in handling syntactically and semantically intricate sentences.

4.7.3.2 Continuous Updates to CNL Concepts

As CNL is a relatively new concept, the system's components, such as the Mapper for POS tagging and karaka relations, required continuous updates. These updates were essential to keep the system aligned with evolving linguistic theories and practices, ensuring accurate representation and tagging.

4.7.3.3 Reliance on Standard Tools

The system heavily relied on current standard tools. When these tools encountered difficulties with certain sentences, it increased the manual workload for linguists. In such cases, linguists often preferred to directly annotate the sentence in the CNL text section rather than relying on the preceding sections. This reliance exposed the limitations of the existing tools and underscored the need for more robust and reliable linguistic tools.

4.7.4 Future Work

4.7.4.1 Support for Complex Sentences

Future work should focus on enhancing the system's ability to support complex sentences. This involves improving the algorithms and tools used for tagging and parsing complex syntactic structures, ensuring that the system can handle a wider variety of sentence constructions with minimal manual intervention.

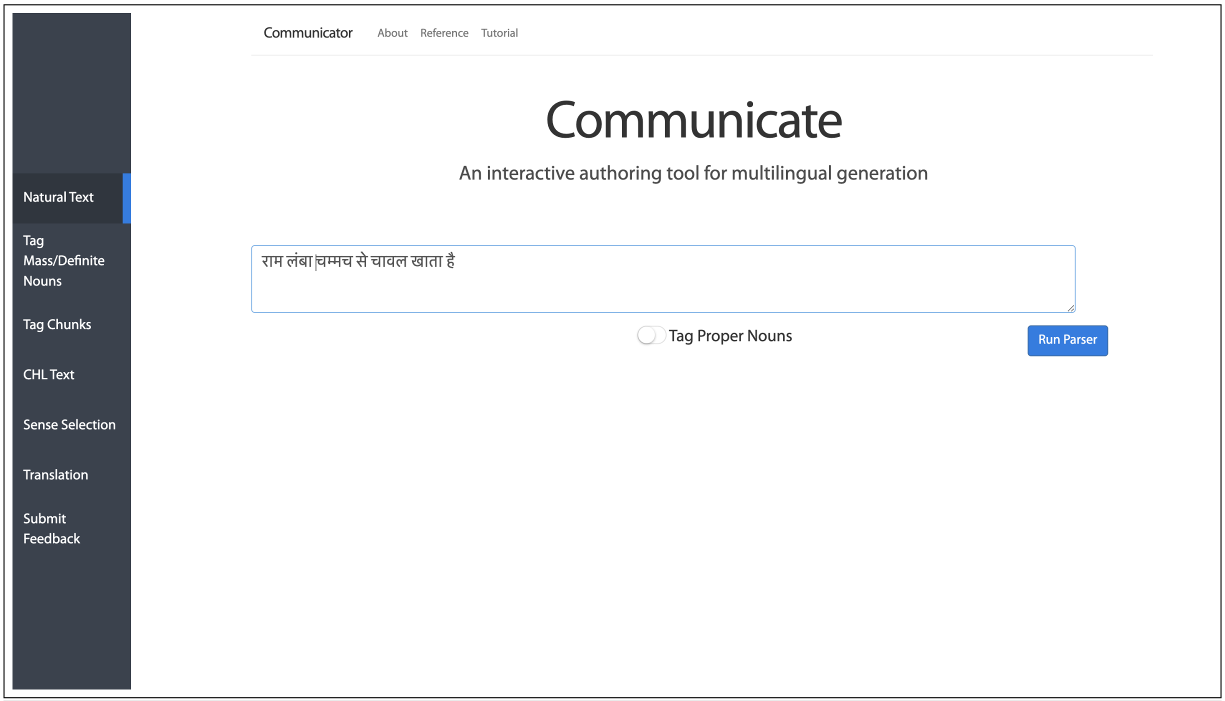
[1] https://github.com/harshavardhangsv/communicate

4.7.4.2 User Interface Enhancements

The current interface should be redesigned to cater to different user needs. One approach is to divide the interface into separate screens tailored for simple and complex use cases. This division can help create a more intuitive and efficient workflow for different types of users:

1. **Simple Users**: Users who are familiar with the language but may not have deep linguistic expertise can be provided with a simplified interface. This interface would streamline the process for straightforward sentences, minimizing the need for manual corrections.

2. **Linguists**: For users with linguistic expertise, a more detailed and flexible interface can be designed. This interface would allow linguists to easily handle complex sentences and provide detailed annotations, leveraging their expertise to refine the CNL text.

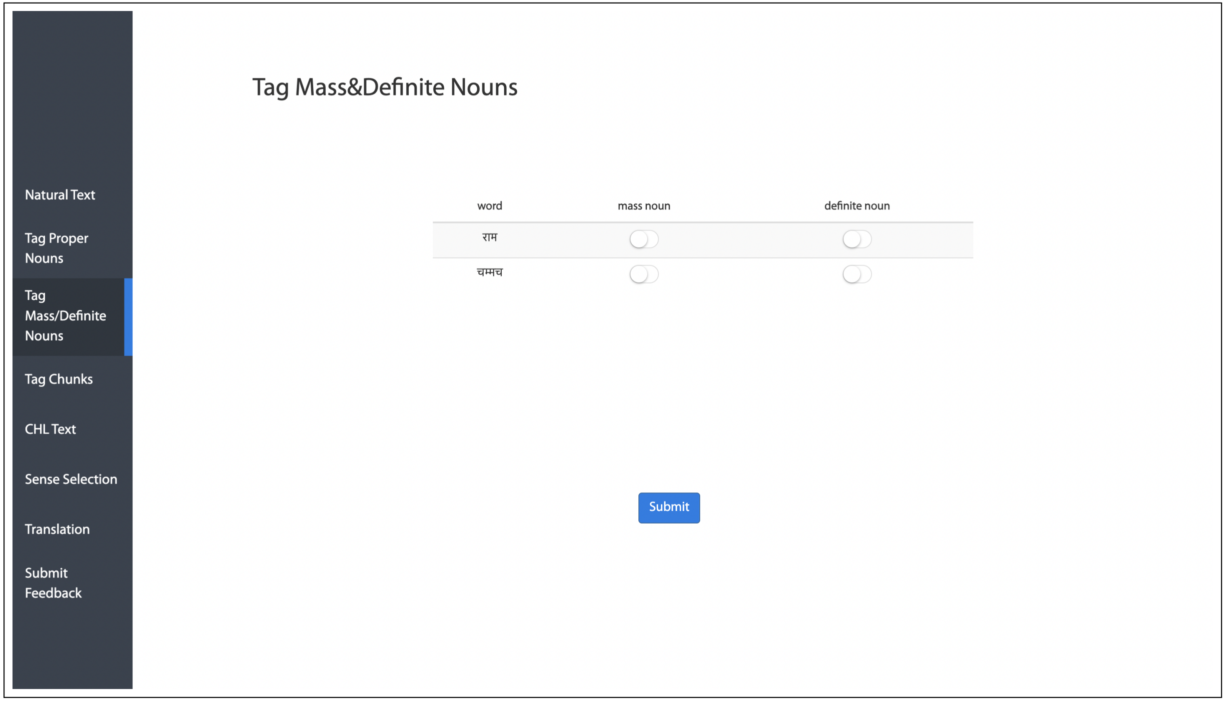


Communicate

An interactive authoring tool for multilingual generation

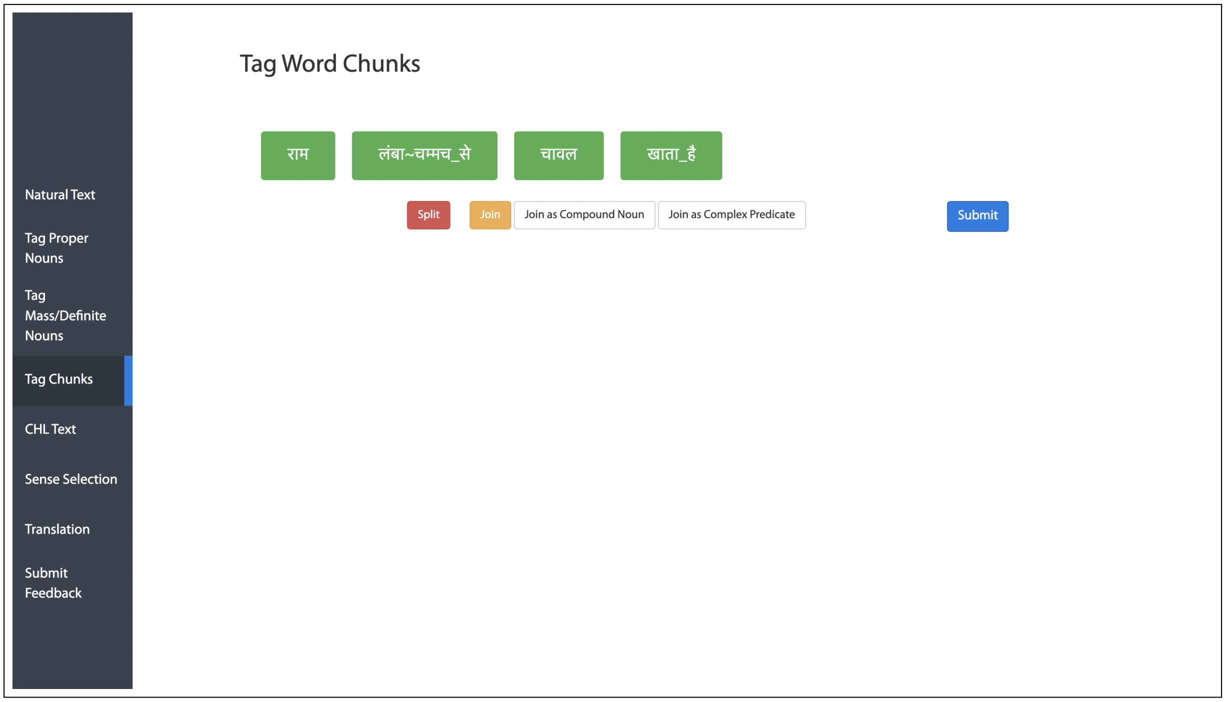
राम लंबा चम्मच से चावल खाता है

○ Tag Proper Nouns

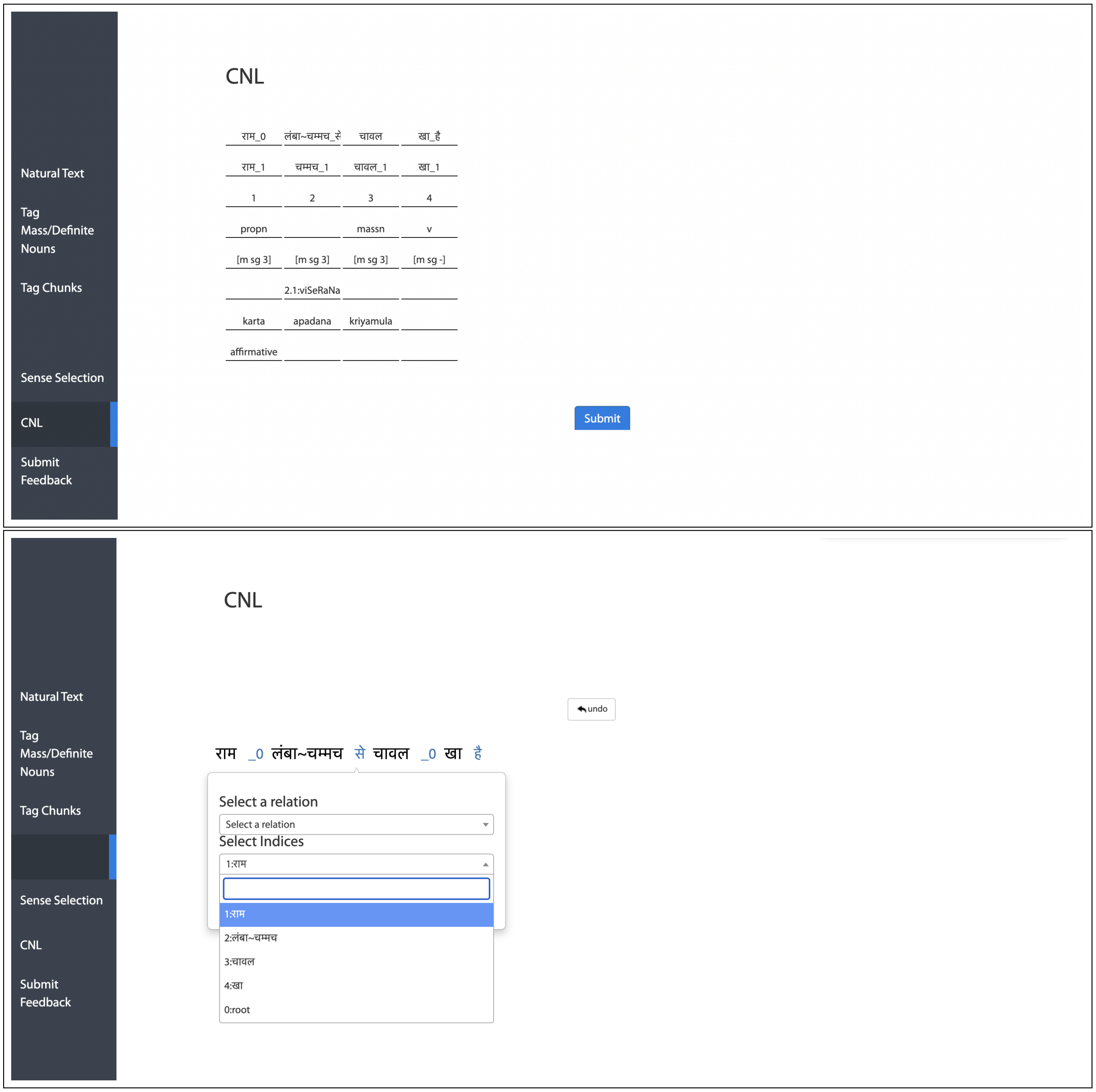


Tag Mass&Definite Nouns

word	mass noun	definite noun
राम	○	○
चम्मच	○	○



Natural Text
Tag Proper Nouns
Tag Mass/Definite Nouns
Tag Chunks
CHL Text
Sense Selection
Translation
Submit Feedback
Tag Word Chunks
राम
लंबा~चम्मच_से
चावल
खाता_है
Split
Join
Join as Compound Noun
Join as Complex Predicate
Submit



CNL

राम_0	लंबा~चम्मच_से	चावल	खा_है
राम_1	चम्मच_1	चावल_1	खा_1
1	2	3	4
propn		massn	v
[m sg 3]	[m sg 3]	[m sg 3]	[m sg -]
	2.1:viSeRaNa		
karta	apadana	kriyamula	
affirmative			

CNL

राम _0 लंबा~चम्मच से चावल _0 खा है

Select a relation

Select a relation

Select Indices

1:राम

1:राम
2:लंबा~चम्मच
3:चावल
4:खा
0:root

Figure 4.2: Screenshots of Communicate Tool

Chapter 5

Conclusion and Future Work

The research presented in this thesis has made significant strides in the development of tools for linguistic and semantic resource development in the context of natural language processing. Through the construction of a knowledge base for user-oriented documents in the AC manual domain, the automated conversion of Indian language morphological processors into the Grammatical Framework, and the creation of an intelligent interactive editor for Controlled Natural Language (CNL) for Hindi text, we have addressed critical challenges in the field. These projects have not only enhanced our understanding of the complexities involved in linguistic resource creation but also provided practical solutions that contribute to the broader NLP community.

Despite the progress made, there are several areas for future research and development. One key area is the refinement of semantic similarity tools for better concept disambiguation in the knowledge base. Improving these tools will enhance the accuracy and usability of the knowledge base, making it more effective for users seeking specific information. Additionally, further work is needed to customize the depth of the ontology to better meet user-centric requirements, ensuring that the level of detail provided is both useful and accessible to end-users.

Another important direction for future work is the expansion of the current methodologies to encompass a broader range of technical domains and languages. This includes extending the PurposeNet framework to other types of user manuals and technical documents, as well as adapting the automated morphological conversion tools for additional Indian languages and other language families. Moreover, the intelligent interactive editor can be further enhanced with more sophisticated user interface features and support for complex sentence structures, thereby improving its usability and effectiveness in real-world applications. By addressing these future research directions, we can continue to advance the field of linguistic and semantic resource development, ultimately contributing to more robust and versatile NLP systems.

Related Publications

- Harsha Vardhan Grandhi, Soma Paul: “Automatic conversion of Indian Language Morphological Processors into Grammatical Framework (GF)”, in proceedings of the 12th International Conference on Natural Language Processing, ICON 2015, Trivandrum, India

Bibliography

[1] R. Ahmad, P. Gupta, N. Vuppala, S. K. Pathak, A. Kumar, G. Soni, S. Kumar, M. Shrivastava, A. K. Singh, A. K. Gangwar, P. Kumar, and M. K. Sinha. Transzaar: Empowers human translators. In *2018 18th International Conference on Computational Science and Applications (ICCSA)*, pages 1–8, 2018.

[2] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider. Abstract meaning representation for sembanking. In *LAW@ACL*, 2013.

[3] D. M. Cer, M. T. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *International Workshop on Semantic Evaluation*, 2017.

[4] A. A. Copestake, D. Flickinger, C. Pollard, and I. A. Sag. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3:281–332, 2005.

[5] G. Détrez, V. M. Sánchez-Cartagena, and A. Ranta. Sharing resources between free/open-source rule-based machine translation systems: Grammatical framework and apertium. In N. Calzolari, K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4394–4400, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).

[6] M. L. Forcada, M. Ginestí-Rosell, J. Nordfalk, J. O'Regan, S. Ortiz-Rojas, J. A. Pérez-Ortiz, F. Sánchez-Martínez, G. Ramírez-Sánchez, and F. M. Tyers. Apertium: a free/open-source platform for rule-based machine translation. volume 25, pages 127–144, 2011.

[7] N. E. Fuchs, U. Schwertel, and R. Schwitter. Attempto controlled english - not just another logic specification language. In *International Workshop/Symposium on Logic-based Program Synthesis and Transformation*, 1998.

[8] M. Humayoun and A. Ranta. Developing punjabi morphology, corpus and lexicon. In *Pacific Asia Conference on Language, Information and Computation*, 2010.

[9] I. Jena, S. Chaudhury, H. Chaudhry, and D. M. Sharma. Developing oriya morphological analyzer using lt-toolbox. In *International Conference on Interaction Sciences*, 2011.

[10] J.-U. Kietz, R. Volz, and A. Maedche. Extracting a domain-specific ontology from a corporate intranet. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*, 2000.

[11] P. Kiran Mayee, R. Sangal, and S. Paul. Automatic extraction and incorporation of purpose data into purposenet. In *2010 2nd International Conference on Computer Engineering and Technology*, volume 6, pages V6–154–V6–158, 2010.

[12] H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen. The protégé OWL plugin: An open development environment for semantic web applications. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *The Semantic Web - ISWC 2004: Third International Semantic Web Conference,Hiroshima, Japan, November 7-11, 2004. Proceedings*, volume 3298 of *Lecture Notes in Computer Science*, pages 229–243. Springer, 2004.

[13] M. Li, X.-Y. Du, and S. Wang. Learning ontology from relational database. In *2005 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3410–3415 Vol. 6, 2005.

[14] S. P. Mohanty, N. J. Wani, M. Srivastava, and D. M. Sharma. Kathaa: A visual programming framework for NLP applications. In J. DeNero, M. Finlayson, and S. Reddy, editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 92–96, San Diego, California, June 2016. Association for Computational Linguistics.

[15] A. Ranta. A revised version of the on-line gf tutorial, v1.0. *A. Beckmann and N. Preining Editors, ESSLLI Course Material I*, 2003.

[16] A. RANTA. Grammatical framework. volume 14, page 145–189, 2004.

[17] A. Ranta. From semantics to computer science: Grammars as software libraries. 2009.

[18] A. Ranta. The gf resource grammar library. volume 2, Dec. 2009.

[19] R. Sangal, S. Paul, and P. K. Mayee. Purposenet: A knowledge base organized around purpose. In H. D. Pfeiffer, D. I. Ignatov, J. Poelmans, and N. Gadiraju, editors, *Conceptual Structures for STEM Research and Education, 20th International Conference on Conceptual Structures, ICCS 2013, Mumbai, India, January 10-12, 2013. Proceedings*, volume 7735 of *Lecture Notes in Computer Science*, pages 29–30. Springer, 2013.

[20] C. Singh and R. Srivastav. Study and population of artifacts. In *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, 2011.

[21] E. Sneiders. Automated faq answering: Continued experience with shallow language understanding. 1999.

[22] S. Tool. Sampark: Machine translation system among indian languages, 2009, [online]. 2009.

[23] C. Zinn and B. Campbell. Weblicht-batch - a web-based interface for batch processing large input with the weblicht workflow engine. In *CLARIN Annual Conference*, 2022.