Active and Passive Voice Conversion : An NLP Application using Prolog

Anurag Shekhar [a] Khushboo Saraf [b] Riddhi Shah [c] Shreyas Kishorkumar Patel [d]

[abcd] Masters of Applied Computer science, Concordia University, Montreal, Canada

August 13, 2022

Abstract

The main paper[1] proposes a solution to one of the critical aspects of English grammar, Active and Passive Voice conversion. The proposed solution is an NLP application, which uses Definite Clause Grammar in Prolog. Although their structures are different, an active statement and its counterpart passive sentence actually reflect the same idea. These structures clarifies how voice demonstrates connections between the verb and the subject and/or object. The sentences are represented by "parse tree" based on a defined grammar. Each part of the sentence is mapped to a part of speech determined by the grammar. Thereafter, the sentence is converted to passive if the input is in active voice or vice versa. The main paper[1] also proposes a 'Auxiliary-based' solution to handle all of the 12 English tenses. In this paper, the main paper[1] is reviewed and implemented using Prolog and JPL. The implementation will cover definition of grammar using DCG, conversion between different voices (i.e active to passive or vice versa) of English sentences. A Java based GUI will also be implemented for making the implementation more user friendly. The implementation of the voice conversion system can be further used in Question-Answering Model and Chat-Bots.

1 Introduction

English sentences can be written in active voice or passive voice. The active voice is a sentence structure that highlights the person who is performing an action. On the other hand, in the passive voice, the action being performed is emphasized. The sentences can be decomposed into several parts according to grammar rules, which forms the base of conversion of Natural language to a form which machine can understand and operate upon Natural language is ambiguous for computers to understand. The interchangable structure of active and passive sentences can help in reducing

ambiguity and make the language processing by machines more efficient. The term paper describes the implementation of conversion logic from active voice to passive voice or vice versa using Natural Language Processing and Definite Clause Grammar. The implementation is majorily based on Prolog with user interface handled by JPL in java.

"Natural language processing strives to build machines that understand and respond to text or voice data and respond with text or speech of their own in much the same way humans do."**-IBM** NLP helps computers communicate with humans in their own language. It helps computers to read text, hear speech, interpret it, measure sentiment and determine which parts are important. Few popular applications of NLP are Predictive text, Language Translation, Speech recognition, digital assistants and so on.

A **Definite Clause Grammar** (DCG) which is a way of expressing grammar, either or natural or formal languages, in a logic programming language such as Prolog. A DCG rule has the form: Head –> Body. A rule's body consists of terminal, nonterminals, and grammar goals. In Prolog, DCG allows the direct implementation of formal logic.

The implementation is based on DCG in prolog and deals with conversion between active sentence to passive sentence using Natural Language Processing grammar. There are 12 Tenses that has been categorized in 4 groups based on presence of auxiliary verbs. This reduces the code verbosity of the whole system and handles all cases effectively. The implementation covers only group 1 which deals with simple present and simple past tense.

2 Literature review

Following are some related works in the similar fields, which has helped in the implementation of the conversion logic :

2.1 An introduction to language processing with perl and prolog - Pierre M Nugues

The transformation rules are defined to describe conversion rules where active/passive sentences are converted to new sentences with a derived phrase structure using some syntactic relation. Tree-to-tree mapping is used to represent the active/passive transformation rule.

2.2 An Approach towards Implementation of Active and Passive voice using LL(1) Parsing - Muhammad Fahad, Hira Beenish

This paper aims at converting active into passive sentences using LL parsing which is a top-down parser and a subset of CFL. They have implemented the same using POS tagging and speech recog-

nition to make it more efficient.

2.3 Student Modelling in an Intelligent Tutoring System for the Passive Voice of English Language: Maria Virvou, Dimitris Maras and Victoria Tsiriga

This paper introduces a new concept of Intelligent tutoring system for Greek students to model passive voice of English language. It focuses on student's diagnosis process which helps student to diagnose the underlying misconception of the their mistake during their session.

3 Design and Architecture

Most of the active voice sentences can be converted to passive voice sentences. Given a sentence with an object, the conversion has three basic rules (Figure 1):

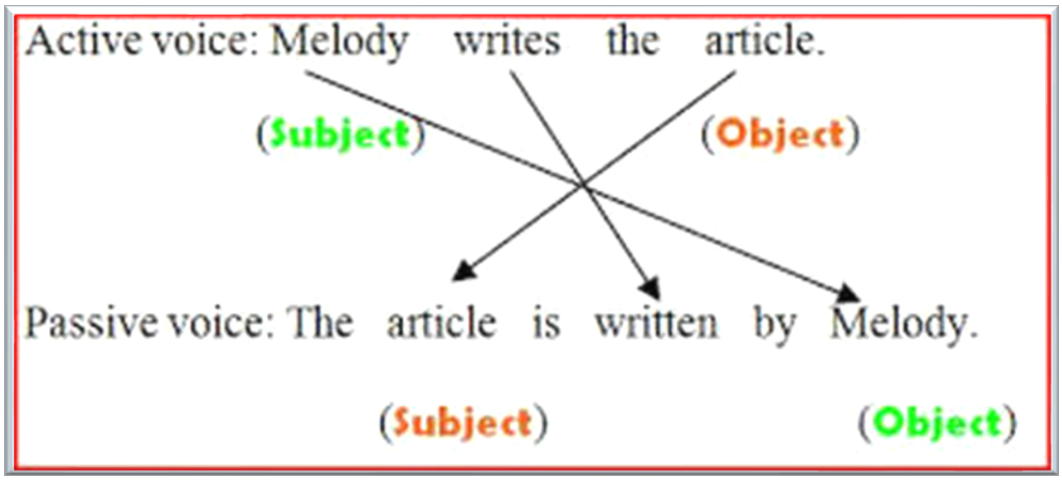

Figure 1: Basic Conversion Rule

1. The object of the active sentence becomes the subject of the passive sentence.
2. The subject of the active sentence becomes the object of the passive sentence.
3. The Finite form of the verb is changed to "to be + past participle".

3.1 Parts of Speech Representation

Representing the structure of a sentence allows us to see the beginnings of semantic relationships between words. Ideally, with the help of these relationships , a representation can be formed which could be used computationally. This representation can be described as parts of speech POS (Figure 2).

As depicted in Figure 2, different parts of speech are as follows :

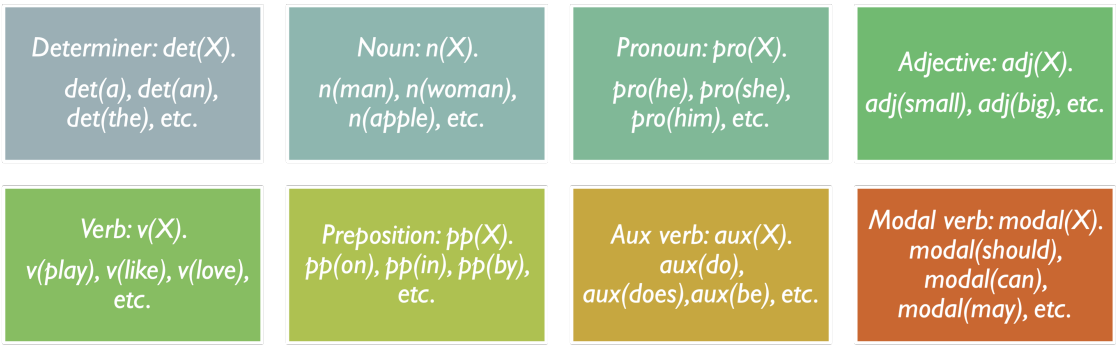

Figure 2: POS Representation

- Determiner
- Noun
- Pronoun
- Adjective
- Verb
- Preposition
- Auxiliary Verb
- Modal Verb

3.2 Grammar

Sentences in natural language are made up of words or phrases, adhere to grammar rules, and express all the semantic information necessary. We can write simple grammars using the DCG notation that recognise if a string of words (represented as a list of atoms) belongs to the language. The different parts of sentence can be tagged to specific parts of speech, which is also known as POS tagging.

An input sentence can be represented as a parse tree with the help of grammar shown in Figure 3. A sentence (S) always has a noun phrase (NP) and verb (PP). Using the grammar rules, the noun and verb phrase can be further decomposed until every part of sentence is pos tagged.

Similarly, given a set of parts of speech elements, a sentence can be generated.

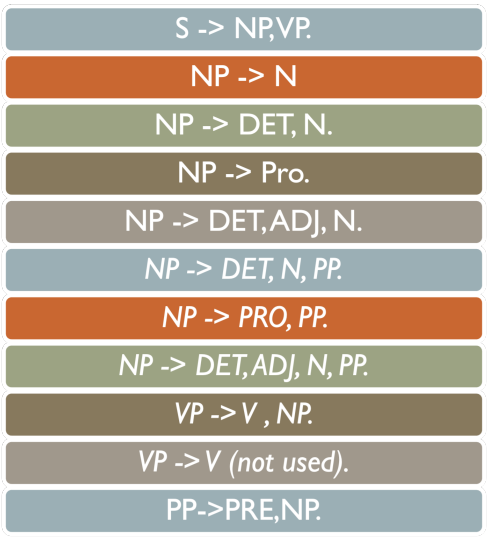

Figure 3: Grammar

3.3 Auxiliary Based Solution

The main paper[1] discusses a auxiliary verb based solution for handling 12 different tenses. The 12 tenses have been divided into group of 4 based on count of auxiliary verbs present in a sentence belonging to a specific sentence.

Figure 4: Auxiliary Based Solution

The Figure 4 shows the division of all 12 tenses in a total of 4 groups as below :

- Group 1: Auxiliary Verb = 0. Tenses : simple past tense and simple present tense.
- Group 2: Auxiliary Verb = 1. Tenses : simple future tense, continuous past tense, continuous present tense, perfect past tense and perfect present tense.
- Group 3: Auxiliary Verb = 2. Tenses : continuous future tense, perfect future tense, perfect continuous past tense, and perfect continuous present tense.

- Group 4: Auxiliary Verb = 3. Tense : perfect continuous future tense.

3.4 Active Voice Sentence Representation

Based on the group determined by auxiliary based solution and the grammar, the active voice sentences have the following representation :

- Group 1 : NP1,V,NP2
- Group 2 : NP1,aux(Aux1),v(Y),NP2
- Group 3 : NP1,aux(Aux2),aux(Aux3),v(Y),NP2
- Group 4 : NP1,aux(Aux3),aux(Aux4),aux(Aux5),v(Y),NP2

3.5 Passive Voice Sentence Representation

Based on the defined grammar, the passive voice sentences have the following representation :

- Group 1 : NP2,AUX,V,agent(by),NP1
- Group 2 : NP22,aux(Aux),auxTense(AuxTense),v(Y1),agent(by),NP11
- Group 3 : NP22,aux(Aux),aux(Aux1),auxTense(AuxTense),v(Y1),agent(by),NP11
- Group 4 : NP22,aux(Aux),aux(Aux1),aux(Aux2),auxTense(AuxTense),v(Y1),agent(by),NP11

3.6 Conversion Steps

With the above information, the sentences can be converted from active to passive or vice versa using three steps (Figure 5) :

Figure 5: Three Step Conversion

1. Input sentence representation is identified as per the defined grammar (Figure 6).

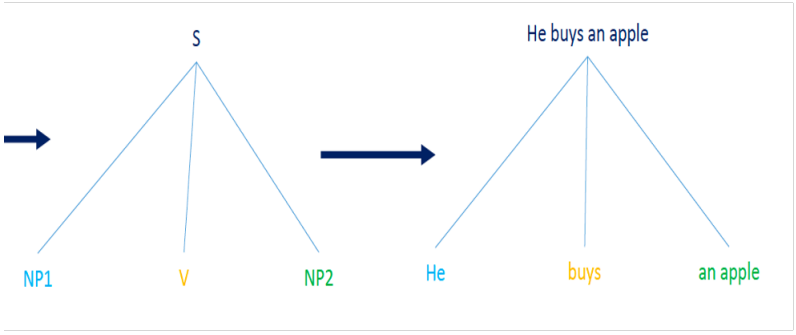

Figure 6: Step 1

2. From the representation of active or passive sentence in step 1, the representation of passive or active sentence is identified respectively. (Figure 7).

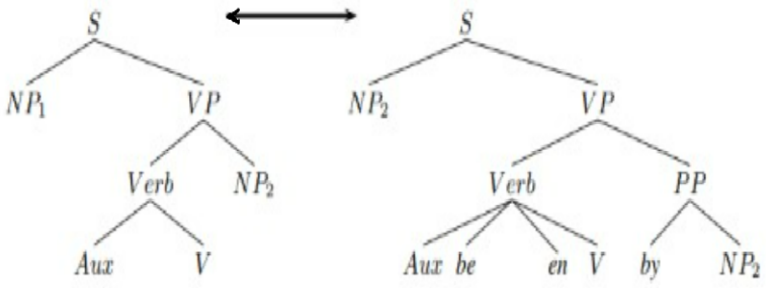

Figure 7: Step 2

3. Output sentence is generated from the representation identified in step 2 (Figure 8).

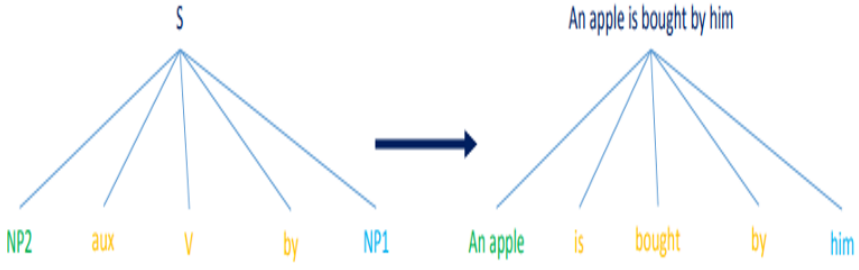

Figure 8: Step 3

Implementation follows the architecture mentioned in (Figure 9).

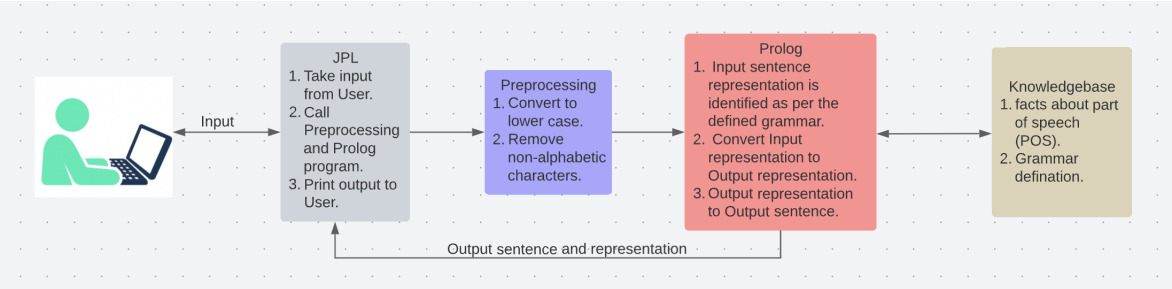

Figure 9: System Architecture

4 Implementation

Group 1 has been implemented based on the solution mentioned in the main paper.

Grammar Definition using DCG (Figure 10).

```
/*========== Noun phrase ==============*/
np(np(DET,N),_,Q) --> det(DET,Q),n(N,Q).
np(np(PRO),P,Q) --> pro(PRO,P,Q).
np(np(DET,ADJ,N),_,Q) --> det(DET,Q),adj(ADJ),n(N,Q).

np(np(DET,N,PP),_,Q) --> det(DET,Q),n(N,Q),pp(PP).
np(np(PRO,PP),P,Q) --> pro(PRO,P,Q),pp(PP).
np(np(DET,ADJ,N,PP),_,Q) --> det(DET,Q),adj(ADJ),n(N,Q),pp(PP).

/*========== Prepositional phrase ==============*/
pp(pp(PRE,NP)) --> pre(PRE),np(NP,object,_).

/*========== Fundamental elements ==============*/
det(det(Word),Q) --> [Word],{lex(Word,det,Q)}.
n(n(Word),Q) --> [Word],{lex(Word,n,Q)}.
pro(pro(Word),P,Q) --> [Word],{lex(Word,pro,P,Q)}.
v(v(Word),Tense,Q,Group) --> [Word],{lex(Word,v,Tense,Q,Group)}.
pre(pre(Word)) --> [Word],{lex(Word,pre)}.

adj(adj([Word])) --> [Word],{lex(Word,adj)}.
adj(adj([Word|L])) --> [Word],{lex(Word,adj)},adj(adj(L)).

agent --> [by].
pol --> [not].
pol(aux(Word),Tense,Q) --> [Word],{lex(Word,pol,Tense,Q)}.
aux(aux(Word),Tense,Q) --> [Word],{lex(Word,aux,Tense,Q)}.
aux(auxTense(Word),Tense) --> [Word],{lex(Word,aux,Tense)}.
aux1(aux(Word),Tense) --> [Word],{lex(Word,aux1,Tense)}.
aux2(aux(Word),Tense) --> [Word],{lex(Word,aux2,Tense)}.
```

Figure 10: Grammar Definition

POS Tagging (Figure 11).

```
/*============================ Active sentence ============================*/

s(s(NP1,V,NP2),Tense,Qs,Qo) -->
        np(NP1,subject,Qs),
        v(V,Tense,Qs,group1),
        np(NP2,object,Qo).

/*========================== Passive sentence ==========================*/

s1(s(NP2,AUX,V,agent(by),NP1),Tense,Qs,Qo) -->
        np(NP2,subject,Qs),
        (
           {\+ NP2=np(pro(i))},
           aux(AUX,Tense,Qs)
        ;
           {NP2=np(pro(i))},
           aux(AUX,Tense,special)
        ),
        v(V,past_participle,Qs,past_participle),
        agent,
        np(NP1,object,Qo).
```

Figure 11: POS Tagging

Conversion Logic (Figure 12).

```
convert(s(NP1,v(Y),NP2),Tense,Qs,Qo,s(NP22,aux(Aux),v(Y1),agent(by),NP11)) :-
    subAndObj(NP1,NP11),
    subAndObj(NP22,NP2),
    (
        \+ NP22=np(pro(i)),
        lex(Aux,aux,Tense,Qo)
    ;
        NP22=np(pro(i)),
        lex(Aux,aux,Tense,special)
    ),
    past_participle(Y,Y1,Qs,Tense).
```

Figure 12: Conversion Logic

The knowledge base has implemented with limited nouns and verbs.

Additionally, pre-processing and user interface has been added using Java. JPL calls are made to the knowledge base which takes an input sentence from user and gives output sentence and its grammar representation to the user.

5 Conclusion

Based on the design and architecture implementation and testing, the following conclusions have been made.

- The implementation can easily use DCG for grammar definition and parts of speech tagging. All the tense cases can be handled where sentences are simple statements. With further modification to design, many other types of sentences can be included like interrogative, affirmative, etc.
- Auxiliary based solution is an effective way to reduce the verbosity in code and handle all the 12 tense cases effectively. Though the implementation has been done for only one of the groups, it can be easily extended for other groups.
- The knowledge base in real time scenario will be very huge and difficult to maintain using prolog. With a complete set of english lexicon which includes all parts of speech and verb forms, known to till date, will result in an enormous knowledge base.
- The main paper does not describe any checks for valid characters in input sentences. In the implementation, few checks have been applied. The checks can be improved using criteria such as ASCII values and more.
- The implemented solution can have a very wide use case in systems like Question-Answering Model Based Systems , Chat Bots, Information Extraction and Sentiment Analysis.

References

[1] Trung Q. Tran , *AandP: Utilizing Prolog for converting between active sentence and passive sentence with three-steps conversion*, (2020)

[2] Pierre M Nugues , *An introduction to language processing with perl and prolog*, (2009)

[3] Muhammad Fahad, Hira Beenish, *An Approach towards Implementation of Active and Passive voice using LL(1) Parsing* , (2020)

[4] Maria Virvou, Dimitris Maras and Victoria Tsiriga , *Student Modelling in an Intelligent Tutoring System for the Passive Voice of English Language* , (2000)

[5] **NLP** : https://towardsdatascience.com/a-simple-nlp-application-for-ambiguity-resolution-bb698d19aff7

[6] **DCG** : https://en.wikipedia.org/wiki/Definite_clause_grammar

[7] **DCG** : https://www.metalevel.at/prolog/dcg

6 Team Members

1. Anurag Shekhar : 40219195
2. Khushboo Saraf : 40204421
3. Riddhi Shah : 40197190
4. Shreyas Kishorkumar Patel : 40192955