CL 1 Project: Question Generation from Fable Stories in Hindi

International Institute of Information Technology, Hyderabad

Gargi Shroff, Swarang Joshi, Ketaki Shetye

Abstract

This document presents a detailed description of the project "Question Generation from Fable Stories" undertaken by Team - DoNut Give Up. The project aims to generate questions from fable stories using dependency grammar and text analysis techniques. It follows a multi-phase approach involving data preprocessing, text analysis, and question formation. The document outlines the project timeline, steps involved in preprocessing, and the methodology for generating questions. The results obtained are also provided, along with additional resources for further exploration.

1 Project Timeline

PHASE 1: Presenting the general outline of the project and procedure to be followed.
PHASE 2: Data Preprocessing and Text Analysis
PHASE 3: Formation of Questions from the Data obtained after Phase 2 using Dependency Grammar

2 Data Preprocessing

The following steps are involved in the preprocessing phase:

2.1 Scraping the Dataset

The required text is scraped from the collected Dataset. The Dataset is obtained from the following resource: Panchatantra Complete Stories (Hindi)

2.2 Removing Punctuation Marks

Punctuation marks are removed from the Dataset to standardize the text data and facilitate analysis. However, the punctuation marks "—" and "," indicating the end of a sentence in Hindi language are not removed.

2.3 POS Tagging

The obtained Dataset is POS tagged to identify the part of speech for each word.POS tagging is used to assign a part-of-speech tag to each word in the dataset. This process helps in understanding the grammatical structure of sentences and the roles played by different words within the fable stories. By tagging each word with its appropriate part of speech, we can analyze the sentence's syntactic properties, such as identifying nouns, verbs, adjectives, and other parts of speech.

This POS tagging is beneficial for question generation as it allows us to identify the key components of the sentences and their relationships. For example, by identifying nouns, we can generate questions about

the subject of the sentence.Adjectives can help generate questions about the characteristics or qualities of characters or objects in the story.

2.4 Dependency Parsing

Dependency parsing is performed on the Dataset to understand the syntactic structure and relationships between words in a sentence.Dependency parsing is a technique used to analyze the syntactic structure of sentences and identify the relationships between words. It helps in understanding how words depend on each other grammatically. Stanza is a Python library commonly used for performing dependency parsing, utilizing deep learning algorithms to accurately parse sentences and extract syntactic information. The output is a parse tree or graph that represents the hierarchical structure of the sentence, with words in form of list of tuples connected by directed arcs indicating their grammatical relationships. It helps in generating meaningful and grammatically correct questions.

3 Dependency Grammar

According to Paninian Grammar, Hindi dependency roles can be explained in terms of Karakas.The head of a sentence is the main verb while the rest of the phrases are children of the main verb. The role of these child nodes with respect to the main verb is the Karaka rols. Let us consider an example for k1 Case (Doer)

Here, generally the subject of the verb is substituted with direct or oblique interrogative words based on whether the word assigned k1 is direct or oblique (direct case does not have postposition between the noun whereas oblique case does).

Example: X (ne) Y ko khaya/kha raha hai.
Question generated : Kaun Y ko kha raha hai ?
Question generated : Kisne Y ko khaya ?
Question generation in our system is often relying on the case of the keyword

4 Question Generation Methodology

4.1 Steps Followed

- After the data preprocessing required for the model (removing the punctuation marks, POS tagging and Dependency Parsing) we iterate through the text and use '—' as a delimiter to mark the end of sentence. Using if else statements we look for words which satisfy the conditions required to form the questions and then replace those words by the appropriate question words.
- We ensure that if multiple questions are possible from a single sentence not all those words are replaced at once from the sentence and all possible questions are formed the sentence.
- Here we look at the possible cases and the questions formed

4.2 Replacing Nominative Case

The word in the nominative case, which is generally the subject of the verb, is replaced with a direct or oblique interrogative word based on the assigned k1 role.

```
if key == 'Case' and value == 'Nom' and (output_list[i][1] == 'NOUN' or output_list[i][1] == 'PRON'):
    if flag == 0:
        if marker == iter:
            temp = "कौन"
            str_question += temp
            str_question += " "
            i = i + 1
            flag = 1
            break
```

4.3 Replacing Accusative Cases

Words in the accusative case, which are usually direct or oblique objects, are replaced with the question word "kise".

```
if key == 'Case' and value == 'Acc' and (output_list[i][1] == 'NOUN' or output_list[i][1] == 'PRON'):
    if flag == 0:
        if marker == iter:
            temp = "किसे"
            str_question += temp
            str_question += " "
            i = i + 1
            flag = 1
            break
```

4.4 Replacing Adjectives

Adjectives with feminine or masculine gender are replaced with "kaisi" and "kaisa" respectively. For Neutral, we have used "kaise".

```
elif output_list[i][1]=='ADJ' and key == 'Gender' and value == 'Fem':
    if flag == 0 :
        if marker == iter :
            temp="कैसी"
            str+=temp
            str+=" "
            i = i+1
            flag = 1
            break
```

```
if output_list[i][1]=='ADJ' and key == 'Gender' and value == 'Masc':
    if flag == 0 :
        if marker == iter :
            temp="कैसा"
            str+=temp
            str+=" "
            i = i+1
            flag = 1
            break
```

```
elif output_list[i][1]=='ADJ':
        if flag == 0 :
            if marker == iter :
                    temp="कैसे"
                    str+=temp
                    str+=" "
                    i = i+1
                    flag = 1
                    break
```

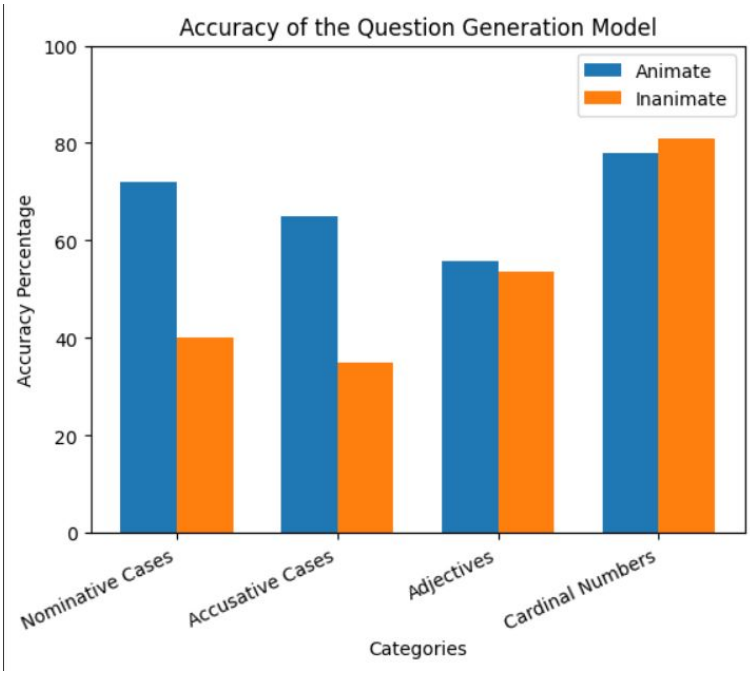
4.5 Replacing Numerals

Words with POS tags indicating numbers are replaced with the question word "kitne".

```
if key == 'NumType' and value == 'Card' and (output_list[i][1] == 'NUM'):
    if flag == 0:
        if marker == iter:
            temp = "कितने"
            str_question += temp
            str_question += " "
            i = i + 1
            flag = 1
            break
```

5 Results Obtained

We have analyzed the questions generated based on the parameters given below:



5.1 Question Generation Analysis

1. The question was generated correctly according to the karaka of the noun identified.
2. The animacy and inanimacy of the noun were identified by the question word.
3. Correct identification for the gender of the adjective is made, and the question is generated accordingly.
4. Questions were formed on the cardinal numbers that were identified.

5.2 Accuracy Percentage

5.2.1 Nominative Case (k1)

- There were 8 stories in the dataset that had animate elements. The accuracy of forming correct questions for these stories for the nominative case was 72
- There were 4 stories in the dataset that had inanimate elements. The accuracy of forming correct questions for these stories for the nominative case was 40
- The overall percentage for the formation of correct questions for the nominative case was 61.33

5.2.2 Accusative Case (k2)

- There were 8 stories in the dataset that had animate elements. The accuracy of forming correct questions for these stories for the accusative case was 65
- There were 4 stories in the dataset that had inanimate elements. The accuracy of forming correct questions for these stories for the accusative case was 35
- The overall percentage for the formation of correct questions for the accusative case was 55

5.2.3 Adjectives

- The overall percentage for the formation of correct questions for adjectives (irrespective of gender) was 54.54

5.2.4 Cardinal Numbers

- The overall percentage for the formation of correct questions for cardinal numbers was 80

6 Analysis

1. The karaka case was accurately identified, and the question words were inserted correctly as required.
2. All possible questions were formed according to the number of possible words belonging to that particular grammatical category in a sentence.
3. Questions were not formed if there was no word belonging to that particular grammatical category in the sentence.

7 Shortcomings

1. The model couldn't differentiate between animate and inanimate objects, resulting in lower accuracy for questions generated for sentences with inanimate objects.
2. When an adjective preceded a noun, the question word was inserted at the position of the noun directly, including the adjective, which led to inaccurate results.

3. The dependency parser couldn't identify other karaka cases like location and instrumental, resulting in the inability to generate questions for these cases.

4. Questions where the question word is inserted at a position different from the original word were not generated correctly.

8 Future Work

1. Incorporate a more accurate dependency parser to generate questions for more grammatical categories accurately, considering other syntactic aspects.

2. Improve question generation accuracy for verbs, which is a challenging part of speech.

3. Utilize WordNet to distinguish between animate and inanimate words, improving the generation of questions accordingly.

4. Expand the dataset and enhance the accuracy of questions generated for complex sentences.

Please note that the generated questions are evaluated for correctness, but there may be cases where the model fails to distinguish between animate and inanimate objects.

9 Resources

- POS Tagger: `https://stanfordnlp.github.io/stanza/pos.html`
- Dependency Parsing: `https://stanfordnlp.github.io/stanza/depparse.html`
- Hindi Question Generation using Dependency Structures: *Radhika Mamidi, Kaveri Anurajama, and Vijjini Rao (LTRC, IIIT Hyderabad)*
  `https://www.academia.edu/66658585/Hindi_Question_Generation_Using_Dependency_Structures`

10 Colab Notebook and GitHub Repo Link

Access our Colab Notebook at: `https://colab.research.google.com/drive/1GBxyUMcwzxwxDij6E_79BGjJipcek18u?usp=sharing`
Access our GitHub Repo at: `https://github.com/jswarang12/QuesGen`

11 Conclusion

While rule-based systems are often considered out of time as compared to machine learning based methods, they don't rely on learning on huge corpora of data unlike the latter. This makes them effective for resource scarce languages.In this project we present a question generation rule set for automated generation of Hindi questions from corresponding sentences in fable stories.

We thank you for exploring the world of question generation using fable stories with us! Should you have any further queries or suggestions, please feel free to reach out to us.