Question Generation from Hindi Stories

Computational Linguistics – 1 | Project 2

Abhinav S Menon (2020114001)
Pratyaksh Gautam (2020114002)
Shashwat Singh (2020114016)

Project 2 कष्टाध्यायी

This project implements the system of question generation proposed in the paper **"Hindi Question Generation Using Dependency Structures"**.

Data Source

For our corpus, we chose the short stories of Premchand, scraped from **this website**.

Methodology

Overview

The data is first sentence tokenized, and questions are removed from the corpus. This is in line with the limitations of the method described in the paper.

Then the data is put through a dependency parser and a list of questions is generated from each sentence, by replacing constituents with appropriate question words.

This list of questions is then printed to an output file.

Algorithm

We have followed the paper's algorithm in most cases; small extensions have been made in the cases of

- Gender agreement for `k1s`: When `k1s` is an adjective, we try to guess its gender from the last matra and alter the gender of the question word कैसा accordingly.
- Proper noun identification for `k5`: When the source is a place with a name (identified using the PoS tag `NNP`, assuming all and only place names are tagged as proper nouns), कहाँ से is used instead of किससे.

- Gender agreement for `r6`: Gender of head noun is identified using the suffix on the possessive, and accordingly changing the gender of the question word किसका.
- `pof` relations: Substituted by क्या in all cases.
- Adjectives, quantifiers and demonstratives for `nmod__adj`: Each case is dealt with separately; replacements are कैसा, कितना and कौनसा respectively. Gender and number are identified using the same strategy as that for `k1s`.
- Coordinate clauses: When a sentence consists of two coordinate clauses (joined by और or तो), questions are generated from each clause separately.
- Subordinate clauses: Relative clauses are deleted before generating the questions.

Important Functions

- `tree.py`
  - `maketree()`: Converts dependency parser output to tree representation.
  - `makesent()`: Converts tree representation to list of words.
- `trav.py`
  - `traverse()`: Generates a list of questions for the whole tree.
- `rules.py`: Contains functions to replace specific constituents with appropriate question words.
- `scraping.py`:
  - `scrape_page()`: Scrapes the page and prints the cleaned data to file.
  - `sentence_tokenize()`: Simply tokenize using regex.

Analysis and Observations

Some discrepancies in the output of the parser lead to correspondingly ill-formed or meaningless questions:

- Sentences were parsed as having multiple roots, especially long and complicated ones.
- Chunking was incorrect for some phrases.
- Quotes were unreliably parsed.
- Relative clauses were not parsed as dependent on the constituent they modify; rather, they are dependent on the verb of the main clause.
- Rhetorical questions occur without a question mark and are therefore not eliminated from the corpus during the tokenisation process (unlike ordinary questions).

Possible Future Versions

Some improvements that can be made to this project in the future are:

- using NER to find place names, person names, etc. (e.g. in order to determine whether to use कौन or क्या to replace `k1` and `k1s`)

- using anaphora to replace pronouns in questions.
- properly handling quoted clauses.
- changing questions to canonical word order.
- eliminating imperative constructions.
- changing the gender of the verb to default (masc).
- eliminating indefinites like कोई, कहीं, etc.