INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY BANGALORE

NATURAL LANGUAGE PROCESSING AI-829

MATHIQ

Aditi Singh
MT2023085
aditi.singh@iiitb.ac.in

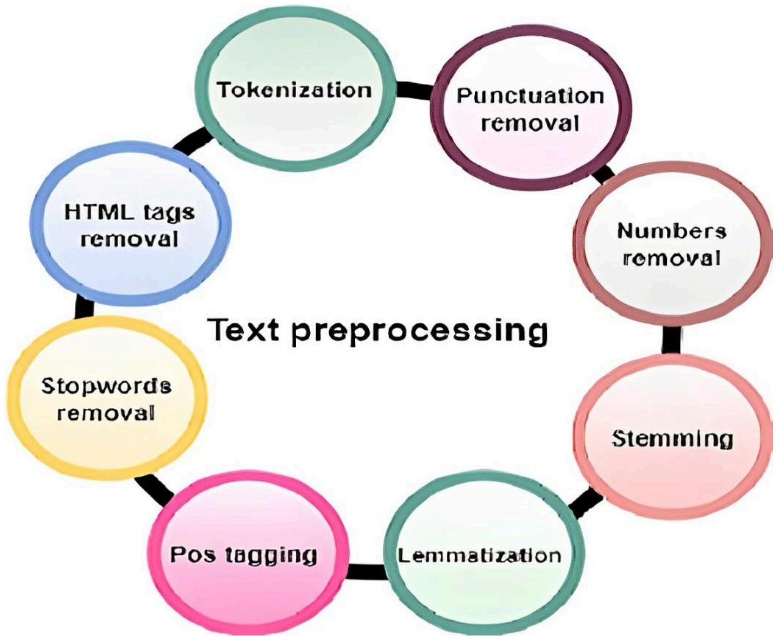
Parag Dutt Sharma
MT2023095
parag.sharma@iiitb.ac.in

MANDATE 2

Introduction

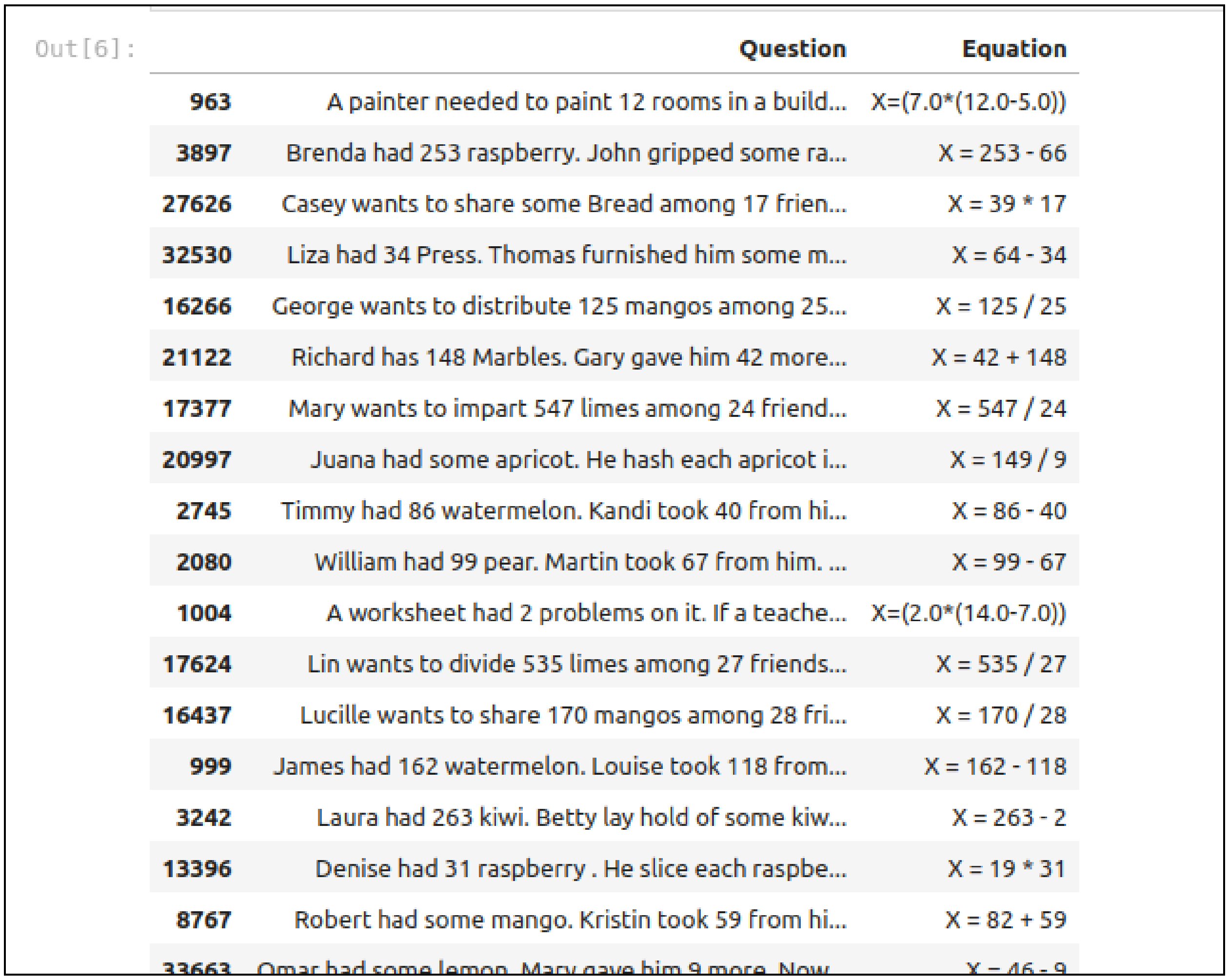
Enhance and streamline the knowledge base by employing essential lexical preprocessing techniques. Initial steps involve preprocessing the dataset by converting text to lowercase, removing unnecessary spaces, and employing tokenization. Subsequently, employ stemming and lemmatization to reduce words to their root forms, promoting better generalization. Identify mathematical statements using robust methods such as CAP, Pointwise Mutual Information (PMI), and N-grams analysis for comprehensive understanding. Recognize diverse problem statement variants to ensure a more inclusive knowledge representation.

Additionally, integrate phonetic hashing to enhance search capabilities and retrieval efficiency. Implement Zero Padding for equalizing sequence lengths, allowing seamless integration into models that require fixed-size input. Convert processed text into tensor sequences, facilitating efficient numerical representation. This holistic approach, coupled with padding spaces and other preprocessing steps, enhances the knowledge base's quality, ensuring a more accurate and standardized information repository.



Dataset

The dataset comprises mathematical word problems, predominantly featuring algebraic questions and quantitative aptitude queries. These scenarios are accompanied by their respective equations, where the imaginary variable 'x' symbolizes the sought-after value.



Out[6]:

	Question	Equation
963	A painter needed to paint 12 rooms in a build...	X=(7.0*(12.0-5.0))
3897	Brenda had 253 raspberry. John gripped some ra...	X = 253 - 66
27626	Casey wants to share some Bread among 17 frien...	X = 39 * 17
32530	Liza had 34 Press. Thomas furnished him some m...	X = 64 - 34
16266	George wants to distribute 125 mangos among 25...	X = 125 / 25
21122	Richard has 148 Marbles. Gary gave him 42 more...	X = 42 + 148
17377	Mary wants to impart 547 limes among 24 friend...	X = 547 / 24
20997	Juana had some apricot. He hash each apricot i...	X = 149 / 9
2745	Timmy had 86 watermelon. Kandi took 40 from hi...	X = 86 - 40
2080	William had 99 pear. Martin took 67 from him. ...	X = 99 - 67
1004	A worksheet had 2 problems on it. If a teache...	X=(2.0*(14.0-7.0))
17624	Lin wants to divide 535 limes among 27 friends...	X = 535 / 27
16437	Lucille wants to share 170 mangos among 28 fri...	X = 170 / 28
999	James had 162 watermelon. Louise took 118 from...	X = 162 - 118
3242	Laura had 263 kiwi. Betty lay hold of some kiw...	X = 263 - 2
13396	Denise had 31 raspberry . He slice each raspbe...	X = 19 * 31
8767	Robert had some mango. Kristin took 59 from hi...	X = 82 + 59
33663	Omar had some lemon. Mary gave him 9 more. Now...	X = 46 - 9

Lexical Preprocessing

- **Lowercase Alphabets:**

  Lowercasing is a fundamental step in NLP data preprocessing, ensuring uniformity by converting all characters to lowercase. This prevents the model from treating the same word differently based on its case, improving overall consistency and generalization.

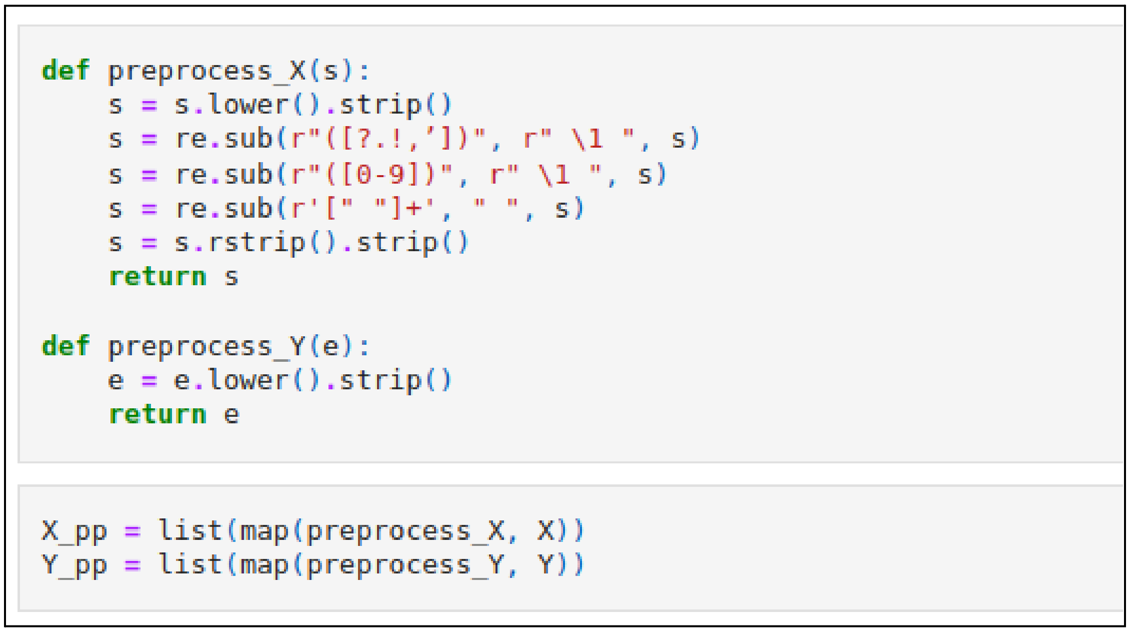
  ‘John has 20 apples.’ ⟶ ‘john has 20 apples.’

- **Padding Spaces and Removing Extra White Spaces:**

  Proper spacing is crucial for effective tokenization. Padding spaces involve ensuring consistent spacing between words, while removing extra whitespaces enhances data cleanliness. This process maintains the integrity of the text, optimizing subsequent tokenization and analysis.

  ‘john has 20 apples.’ ⟶ ‘john has 2 0 apples .’

  ‘X = (7.0 ∗ (12.0 − 5.0))’ ⟶ ‘x = (7 . 0 ∗ (1 2 . 0 − 5 . 0))’



```python
def preprocess_X(s):
    s = s.lower().strip()
    s = re.sub(r"([?.!,’])", r" \1 ", s)
    s = re.sub(r"([0-9])", r" \1 ", s)
    s = re.sub(r'[" "]+', " ", s)
    s = s.rstrip().strip()
    return s

def preprocess_Y(e):
    e = e.lower().strip()
    return e
```

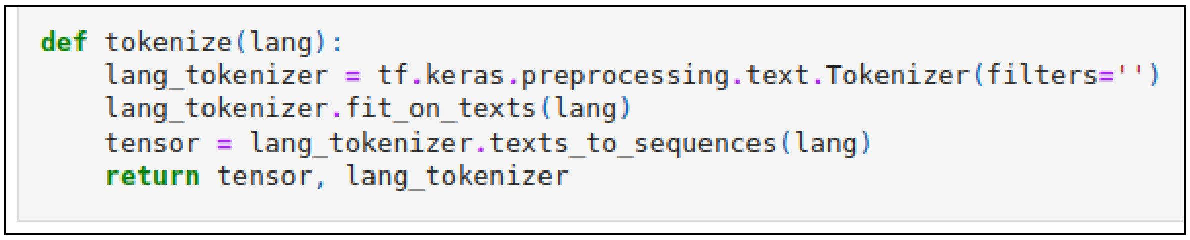
```python
X_pp = list(map(preprocess_X, X))
Y_pp = list(map(preprocess_Y, Y))
```

- **Tokenization:**

  Tokenization involves breaking down a text into smaller units, typically words or subwords. This step is crucial in NLP as it provides the model with discrete elements to analyze. Tokens serve as the basic building blocks for subsequent processing, facilitating efficient feature extraction and analysis.

  ‘john has 2 0 apples .’ → [‘john’, ‘has’, ‘2’, ‘0’, ‘apples’, ‘.’]

  ‘x = (7 . 0 ∗ (1 2 . 0 − 5 . 0))’ → [‘x’, ‘=‘, ‘(‘, … , ’)’]



```
def tokenize(lang):
    lang_tokenizer = tf.keras.preprocessing.text.Tokenizer(filters='')
    lang_tokenizer.fit_on_texts(lang)
    tensor = lang_tokenizer.texts_to_sequences(lang)
    return tensor, lang_tokenizer
```

- **Converting to Tensor Sequences:**

  Converting text data to tensor sequences is essential for numerical representation in NLP models. Tensors are mathematical entities that can be efficiently processed by machine learning algorithms. This conversion enables the model to work with structured numerical data, enhancing its ability to learn and generalize patterns.

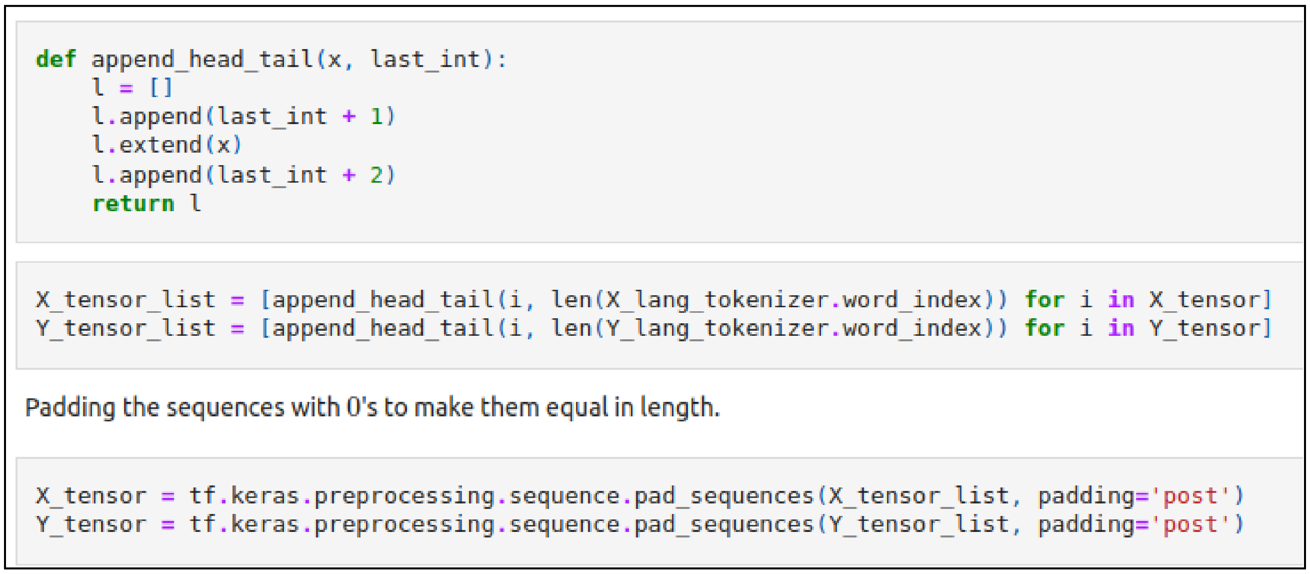
  [‘john’, ‘has’, ‘2’, ‘0’, ‘apples’, ‘.’] → [104, 2454, … , 69, 911]

  [‘x’, ‘=’, ‘(’, … , ‘)’] → [2,1, … ,6]

- **Zero Padding for Equalizing Sequence Lengths:**

  In NLP, sequences of varying lengths can pose challenges for model training. Zero padding involves adding zeros to shorter sequences, ensuring equal lengths. This is crucial for the successful implementation of neural networks, enabling consistent input dimensions and preventing issues related to variable sequence lengths during model training.

  [104, 2454, … , 69, 911] → [104, 2454, … , 69, 911, 0, … , 0, 0, 0]



```
def append_head_tail(x, last_int):
    l = []
    l.append(last_int + 1)
    l.extend(x)
    l.append(last_int + 2)
    return l
```

```
X_tensor_list = [append_head_tail(i, len(X_lang_tokenizer.word_index)) for i in X_tensor]
Y_tensor_list = [append_head_tail(i, len(Y_lang_tokenizer.word_index)) for i in Y_tensor]
```

Padding the sequences with 0's to make them equal in length.

```
X_tensor = tf.keras.preprocessing.sequence.pad_sequences(X_tensor_list, padding='post')
Y_tensor = tf.keras.preprocessing.sequence.pad_sequences(Y_tensor_list, padding='post')
```

Milestones To Achieve In Future Mandates

Mandate 3: Utilizing Shallow Parsing and POS Tagging helps establish required grammatical structures in problem statements. The usage of Hidden Markov Models (HMMs) with the Viterbi Heuristic allows for effective modeling of word sequences. This approach helps in understanding the correlation of mathematical terms and managing variations in problem presentation. We will start with using models like BERT, GPT, and LAMA 2.

Mandate 4: Implement Word Sense Disambiguation (WSD) techniques to accurately understand the meanings of ambiguous words in specific mathematical contexts. Integrate Named Entity Recognition (NER) to identify and categorize entities like mathematical terms, symbols, and units in given math problems. Apply Topic Modeling to capture the underlying mathematical terms in user queries. Employ fine-tuning methods and design a user-friendly interface to optimize the performance of MathIQ.

Team Members and Their Role

ADITI SINGH (MT2023085):

Focuses on the technical aspects of model training and optimization.

Handles tasks like:

- Data pre-processing and curation.
- Choosing and configuring the deep learning model architecture.
- Training and hyperparameter tuning of the model.
- Evaluating the model's performance and analyzing errors.

PARAG DUTT SHARMA (MT2023095):

Focuses on the natural language processing and symbolic representation aspects.

Handles tasks like:

- Developing the NLP pipeline for text understanding and problem extraction.
- Implementing the LaTeX encoding and processing functionalities.
- Designing the output presentation format (LaTeX, natural language, or hybrid).
- Creating user interface elements for problem input and solution display.