ParaphraseDatasetCreation Documentation

Shubham Nemani, Pooja Verma, Anish M M

Dec 12, 2020

CONTENTS:

CHAPTER

ONE

PARAPHRASEDATASETCREATION

1.1 ParaphraseDatasetCreation package

1.1.1 Subpackages

ParaphraseDatasetCreation.Hindi package

Submodules

ParaphraseDatasetCreation.Hindi.Negative module

class ParaphraseDatasetCreation.Hindi.Negative.**Negative_Paraphrases**
Bases: object

This class generates negative paraphrase for input sentence using 2 methods: 1. proper_noun_swap() 2. negation_by_compound()

get_dep_tree(*text*)
function to do dependency parsing of sentence

Parameters **text** (*string of words*) – sentence to be parsed

Returns list of tuples of form (parent word , dependency relation , current word)

Return type list of tuples

get_tagged_sent(*text*)

function to do POS tagging of sentence by tokenizing it into words return list of words with corresponding POS tag

Parameters **text** (*string*) – string of words to be tagged

Returns list of tuples of form (word,tag)

Return type list of tuples

negation_by_compound(*text*)
function which returns negative paraphrase of given sentence , by swapping Proper Nouns (NNP)

Parameters **text** (*string*) – sentence to be paraphrased

Returns negative paraphrased sentence

Return type string

proper_noun_swap(*text*)
function which returns negative paraphrase of given sentence , by swapping Proper Nouns (NNP)

Parameters **text** (*string*) – sentence to be paraphrased

Returns negative paraphrased sentence

Return type string

ParaphraseDatasetCreation.Hindi.Positive module

class ParaphraseDatasetCreation.Hindi.Positive.**Positive_Paraphrases**
Bases: object

This class generated positive paraphrases using 2 methods: 1. get_paraphrase_by_synonym_change 2. get_paraphrase_by_change_conj

find_pos_tag(*tag*)
it return POS tag on which synset is searched , we are only replacing Noun , Verb and Adjective

Parameters **tag** (*string*) – POS tag

Returns pyiwn object of POS tag

Return type string

get_dep_tree(*text*)
function to do dependency parsing of sentence

Parameters **text** (*string of words*) – sentence to be parsed

Returns list of tuples of form (parent word , dependency relation , current word)

Return type list of tuples

get_paraphrase_by_change_conj(*text*)

function which returns negative paraphrase of given sentence , by changing order of conjunctions
dependency tuple format: (word_parent<index,text> , dep_relation , word_child<index,text>) conj_indx will store indexes of all words modifed as conj (i.e. I like a , b and c) ,so contain index of a , b ,c

Parameters **text** (*string*) – sentence to be paraphrased

Returns paraphrased sentence by changing order of conjunctions

Return type string

get_paraphrase_by_synonym(*text*)
function which returns paraphrase of given sentence , by synonyms substitution it replaces each word with its Synonyms , Proper Nouns are not replaced

Parameters **text** (*string*) – sentence to be paraphrased

Returns paraphrased sentence by synonym substitution

Return type string

get_synonym(*synonyms*, *word*)
input is word and list of its synonyms , return synonym which is different from current word since list of synonyms also contain same word.

Parameters

- **synonyms** (*list*) – list of synonyms for given word
- **word** (*string*) – word for which synonyms to be returned

Returns synonym of given word

Return type string

get_synonyms(*word*, *tag*)
return list of all synonyms for given word according to given POS tag

Parameters

- **word** (*string*) – word for which synonyms to be returned
- **tag** (*string*) – POS tag for given word

Returns list of synonyms of given word

Return type list

get_tagged_sent(*text*)

function to do POS tagging of sentence by tokenizing it into words return list of words with corresponding POS tag

Parameters **text** (*string*) – string of words to be tagged

Returns list of tuples of form (word,tag)

Return type list of tuples

Module contents

ParaphraseDatasetCreation.Malayalam package

Submodules

ParaphraseDatasetCreation.Malayalam.Negative module

This file houses methods for creating negative paraphrase samples for Malayalam. Unlike Hindi, good parsers, Pos-taggers etc are not available for Malayalam yet. So, some methods used for Hindi cannot be used here. Even though IndoWordnet provides synonym functionality, we cannot use tags to select the correct ones and so, that is also avoided since it created a lot of errors.

class ParaphraseDatasetCreation.Malayalam.Negative.**NegativeParaphrases**
Bases: object

This class houses the methods for creating sentence pairs that are not paraphrases. Since we could not find usable open-source packages to find synonyms or get parses for Malayalam, we decided to implement various rule-based approaches. Negative pairs can trivially be created by returning some unrelated sentence or a non-grammatical sentence. But we intend to create grammatical sentences that share same context as the query sentence.

generate(*sent*)
Attempt to generate negative samples using all the implemented methods and return the aggregated list.

Parameters **sent** (*string*) – sentence to get negative paraphrase samples for.

Returns list of negative paraphrase samples.

Return type list of strings

negate_if_last_word_is_is(*sent*)
If the last word of the sentence is '[U+0D06][U+0D23][U+0D4D]' (is), then the sentence can be negated by changing the last word to '[U+0D05][U+0D32][U+0D4D][U+0D32]' (isn't).

Parameters **sent** (*string*) – sentence to get negative paraphrase samples for.

Returns list of negative paraphrase samples.

Return type list of strings

subst_with_non_synonyms(*sent*)
Create negative paraphrase samples by replacing a word with another word from the set of non-synonyms.

Parameters **sent** (*string*) – sentence to get negative paraphrase samples for.

Returns list of negative paraphrase samples.

Return type list of strings

ParaphraseDatasetCreation.Malayalam.Positive module

This file houses methods for creating positive paraphrase samples for Malayalam. Unlike Hindi, good parsers, Pos-taggers etc are not available for Malayalam yet. So, some methods used for Hindi cannot be used here. Even though IndoWordnet provides synonym functionality, we cannot use tags to select the correct ones and so, that is also avoided since it created a lot of errors.

class ParaphraseDatasetCreation.Malayalam.Positive.**PositiveParaphrases**
Bases: object

This class houses the methods for creating paraphrases for a given sentence. Since we could not find usable open-source packages to find synonyms or get parses for Malayalam, we decided to implement various rule-based approaches.

back_translation(*sent*)
Automatic Back Translation can be used to create paraphrases. The paraphrases may not always be correct because translators can make errors. But like the other methods outlined, this can be used to create an auxiliary paraphrase dataset.

Parameters **sent** (*list of strings*) – sentence to be paraphrased

Returns list of paraphrases

Return type list of strings

generate(*sent*)
Attempt to generate positive samples using all the implemented methods for single sentenceand return the aggregated list.

Parameters **sent** (*string*) – sentence to get paraphrase samples for.

Returns list of paraphrase samples.

Return type list of strings

generate_for_pair(*sent1*, *sent2*, *l1*, *l2*)
Attempt to generate positive samples using all the implemented methods for pairs of sentences and return aggregated list.

Parameters

- **sent1** – first sentence in pair
- **sent2** – second sentence in pair

- **l1** (*string*) – language of first sentence in pair
- **l2** (*string*) – language of second sentence in pair

Returns list of paraphrases

Return type list of strings

morphology_and_agglutination_based_paraphrasing(*sent*)

Malayalam is an agglutinative language. This means that words can get clubbed together. But in Malayalam, it is also perfectly valid to write these as separate words. For example: "that [I] saw" can be written as: " [U+0D15][U+0D23][U+0D4D][U+0D1F][U+0D41] [U+0D0E][U+0D28][U+0D4D][U+0D28][U+0D4D] "," [U+0D15][U+0D23][U+0D4D][U+0D1F][U+0D46] "/" [U+0D15][U+0D23][U+0D4D][U+0D1F][U+0D46][U+0D28][U+0D4D][U+0D28][U+0D4D] " or " [U+0D15][U+0D23][U+0D4D][U+0D1F][U+0D41][U+0D35][U+0D46][U+0D28][U+0D4D][U+0D28 " Hence, such cases can be used as a rule-based approach to paraphrasing. These rules ([U+0D38][U+0D28][U+0D4D][U+0D27][U+0D3F] [U+0D28][U+0D3F][U+0D2F][U+0D2E][U+0D19][U+0D4D][U+0D19][U+0D7E] / Sandhi rules) exist in the language and can be coded up. Here we use just one of these rules. Others can also be included similarly.

Parameters **sent** (*list of strings*) – sentence to be paraphrased

Returns list of paraphrases

Return type list of strings

translate_pairs_to_paraphrases(*sent1*, *sent2*, *l1*, *l2*)

Take a pair of sentences in 2 languages l1 and l2. Then convert both sentences into the target language using some pre-existing API. If they are not identical, then they could be paraphrases.

Parameters

- **sent1** – first sentence in pair
- **sent2** – second sentence in pair
- **l1** (*string*) – language of first sentence in pair
- **l2** (*string*) – language of second sentence in pair

Returns list of paraphrases

Return type list of strings

Module contents

1.1.2 Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

S

T