Coreference Resolution

# Technical document

Meghnad

Coreference Resolution Document Version

|  |  |  |
| --- | --- | --- |
| ***Author*** | ***Version*** | ***Date*** |
| *Nayan Sarkar* | *1* | *02-08-2022* |

Contents

[1. Introduction 3](#_Toc110457496)

[2. Prerequisites 3](#_Toc110457497)

[3. Config 3](#_Toc110457498)

[*Configuration* 3](#_Toc110457499)

[4. Coref resolver & sample output 3](#_Toc110457500)

[5. Code Module 5](#_Toc110457501)

[*Folder structure* 5](#_Toc110457502)

[*ResolveCorefConfig* 6](#_Toc110457503)

[ *get\_resolve\_coref\_configs(self)* 6](#_Toc110457504)

[*ResolveCoref* 6](#_Toc110457505)

[ *get\_coref\_resolved(self, seq:str, known\_tags:List[str])* 6](#_Toc110457506)

[6. Conclusion 6](#_Toc110457509)

# Introduction

Coreference Resolution refers to the task of finding all expressions that point to a particular entity in a give text. Generally, pronouns are used to mention different noun forms. In a long text where multiple entities are being mentioned, it is important to know which pronouns are referenced to which entity. Coreference Resolution offers a solution to this problem.

# Prerequisites

Coreference Resolution in this case is an unsupervised method where two python modules named “AllenNLP” and “Coreferee” have been used to build a solution. The “coref\_resolve” module uses pretrained models from both python modules and extracts the best out of them. Thus, for this module to work properly we need above mentioned modules installed properly.

Nevertheless, the text provided for Coreference Resolution should contain both noun(s) and pronoun(s) for the model to work effectively.

# Config

## *Configuration*

The config file contains the following parameters:

1. **tag**: Tags which are to be targeted in coreference resolution.
2. **pos**: Poses which are to be targeted in coreference resolution.
3. **allenNLPurl**: URL from which pretrained model is to be fetched.

# Coref resolver & sample output

Coref resolver module is responsible for coreference resolution based on the config parameters. This module yields entity mentions for each sentence in a text in a dictionary where the key is sentence number and values are list of entities mentioned in that sentence.

The sample output is as follows,

**Input text**: “Maruti Suzuki WagonR is a very good hatchback segment car. It has a decent looks and interior of this car looks nice. This car has good performance and better mileage than Baleno.”

**Known tags**: None

**Output Dictionary**:

{

0:

{

'seq': Maruti Suzuki WagonR is a very good hatchback segment car.,

'tags': ['Maruti Suzuki WagonR']

},

1: {

'seq': Maruti Suzuki WagonR has a decent looks and interior of Maruti Suzuki WagonR looks nice.,

'tags': ['Maruti Suzuki WagonR']

},

2: {

'seq': Maruti Suzuki WagonR has good performance and better mileage than Baleno.,

'tags': ['Maruti Suzuki WagonR', 'Baleno']

}

}

**Input text**: “Maruti Suzuki WagonR is a very good hatchback segment car. It has a decent looks and interior of this car looks nice. This car has good performance and better mileage than Balino.”

**Known tags**: [“Baleno”,”Wagon R”]

**Output Dictionary**:

{

0: {

'seq': 'Wagon R is a very good hatchback segment car.',

'tags': ['Wagon R']

},

1: {

'seq': 'Wagon R has a decent looks and interior of Wagon R looks nice.',

'tags': ['Wagon R']

},

2: {

'seq': 'Wagon R has good performance and better mileage than Baleno.',

'tags': ['Wagon R', 'Baleno']

}

}

# Code Module

## *Folder structure*

## *ResolveCorefConfig*

This class belongs to ‘\resolve\_coref\cfg\config.py’. The details of the functions under this class are as follows,

### *get\_resolve\_coref\_configs(self)*

Returns a dictionary of coref\_resolver config parameters

## *ResolveCoref*

This class belongs to ‘\resolve\_coref\src\coref\_resolver.py’. The details of the functions under this class are as follows,

### *get\_coref\_resolved(self, seq:str, known\_tags:List[str])*

This function takes a sequence(text) and known\_tags (list of strings) as input and yields a dictionary containing sentence index as keys and list of entities as values.

# Conclusion

This module can further be used to infer which entities are mentioned in which sentence in a text and then find corresponding emotion, or to extract other features related to an entity.