ONTOLOGY BASED INFORMATION EXTRACTION AND ENRICHMENT FOR PURCHASE DOCUMENTS

A thesis Submitted in partial fulfillment of the requirements for the award of the degree of

B.Tech

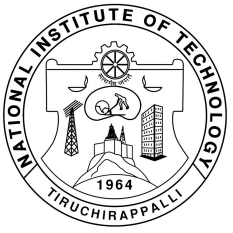
In

ELECTRICAL AND ELECTRONICS ENGINEERING

By

RAJ KRISHNAN. V (107115073)

RAVIKIRAN. R (107115076)



DEPARTMENT OF
ELECTRICAL AND ELECTRONICS ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
TIRUCHIRAPPALLI-620015

MAY 2019

ABSTRACT

Information Extraction aims to retrieve certain types of information from natural language text by processing them automatically. Ontology Based Information Extraction has recently emerged as a subfield of Information Extraction. Here, ontologies - which provide formal and explicit specifications of conceptualizations - play a crucial role in the information extraction process. Ontology is a structured way of representing the relationship between entities, objects and instances and providing contextual sense when dealing with data in a specialised knowledge domain. Development of ontology for a specialised domain can greatly augment information extraction and knowledge representation using natural language processing.

The increase in volume and the need for faster and more accurate processing of product purchase documents has served as an inspiration to perform research in the domain of information extraction. Computer vision and optical character recognition algorithms can perform digitalisation and automation of the text extraction from purchase documents. Development of specialised ontology for electrical products can help extract and enrich technical information with a higher degree of accuracy. Natural Language Processing has been used to complement the work done using Ontology. Feature extraction has been made possible using Natural Language Processing. Hence this thesis deals with information extraction and enrichment of electrical purchase documents with the aid of an ontology.

Keywords : Ontology, Natural Language Processing, Electrical Machines, Computer vision, optical character recognition

ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to the following people for guiding us through this course and without whom this project and the results achieved from it would not have reached completion.

Dr. S. Sudha, Professor and Head of the Department in the Department of Electrical and Electronics Engineering, our project guide, for providing us with the opportunity to work under her on such an interesting project, and allowing us to avail the facilities at the department.

Dr. K. Rajbabu, former research scholar at CSE department of NIT Trichy and employee at BHEL for providing us technical guidance throughout the project and inviting us to his guest lecture on 'Information extraction using Ontology'.

Our hearty thanks to the Department Project Evaluation Committee (DPEC), comprising of **Dr. P. Raja** and **Dr. Josephine R.L** for their valuable suggestions during the reviews.

We are also thankful to the faculty and staff members of the Department of Electrical and Electronics Engineering, our individual parents and our friends for their constant support and help.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

\

IE	-	Information Extraction
NLP	-	Natural Language Processing
OCR	-	Optical Character Recognition
POS	-	Parts of Speech
CV	-	Computer Vision
NLTK	-	Natural Language Toolkit
NER	-	Named Entity recognition
SPARQL	-	SPARQL Protocol and RDF Query Language
API	-	Application Program Interface

CHAPTER 1
INTRODUCTION

1.1 Background

There has been an explosive growth in the amount of information available on networked computers around the world, much of it in the form of natural language documents. Information Extraction is the task of locating and understanding specific pieces of data within a natural language document. Moreover, the advent of the internet has given IE a particular commercial relevance.

IE's ultimate goal, which is the detection and extraction of relevant information from textual documents, depends on proper understanding of text resources. Rule- based IE systems are limited by the rigidity and ad-hoc nature of the manually composed extraction rules. As a result, they present a very limited semantic background.

The role of semantics in IE is often reduced to very shallow semantic labelling. Semantic analysis is considered more as a way to make the linguistic syntax more clear than as a way to build a conceptual interpretation. Today, most of the IE techniques that involve semantic analysis makes use of the simplest part of the whole spectrum of the given domain knowledge, which are known as named entities. However, the growing need for IE application to specialised expert domains such as 'genomics' that require more text understanding pushes towards more sophisticated semantic knowledge resources andthus towards ontologies viewed as conceptual models.

Ontology is a structured way of representing the relationship between entities, objects and instances and providing contextual sense when dealing with data in a specialised knowledge domain. Here, ontologies - which provide formal and explicit specifications

of conceptualizations - play a crucial role in the information extraction process by providing complete conceptual information related to the entities and objects in the given knowledge domain. Therefore, it can greatly augment information extraction and knowledge representation using natural language processing.

Natural language processing (NLP) is a subfield of computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data. As the amount of information available online is increasing day by day, the need to access and process it becomes more important. To convert information from one language to another, machine translation can be used. The NLP techniques are very useful for sentiment analysis. It helps to identify the sentiment among several online posts and comments. Using NLP the Automatic summarization can be performed more efficiently.

1.2 Project overview

In this thesis we tackle the specific problem of using information extraction and enrichment for electrical purchase documents. We use computer vision (CV) and optical character recognition (OCR) to recognise and extract text from purchase tender documents. Due to lack of presence of a suitable ontology for electrical equipments, we have developed a simple ontology describing three electrical products, namely single-phase transformer, three-phase transformer and rheostat. Using the ontology, we have developed a Named Entity recogniser (NER) model which was trained and tested using the product tenders procured from the electrical machines lab of the Department of Electrical and electronics engineering department of NIT Trichy. The ontology was populated with named individuals encountered in the dataset. SPARQL query, a query language used to retrieve information from RDF format ontologies, was used to retrieve information in a structured manner, which was then compared with the dataset to

identify entities and its corresponding attributes. Identification of entities, along with the Named Entity recogniser will provide sufficient information to produce a highly structured output, which can be used for further applications, including development of a centralised purchase system across the institute.

1.3 Project objectives

1. Developing an ontology for electrical machines purchase domain using protege software
2. Performing ontology and NLP based enrichment of the purchase request document with additional information
3. Laying the platform to develop a completely automated process to extract useful information related to description of products from purchase documents for future applications

1.4 Organisation of thesis

The structure of the thesis and various chapters and their descriptions are as below.

- Chapter 1 gives a brief introduction about the background, importance, objectives and scope of the projects
- Chapter 2 presents the literature review conducted to learn about various techniques to develop an ontology, and about
- Chapter 3 describes the areas of focus and the overall flow of our project work
- Chapter 4 deals with the methodology adopted and the specific algorithms used in the project
- Chapter 5 deals with the results obtained, concludes the thesis and presents the future scope of work.

CHAPTER 2
LITERATURE REVIEW

1. Natural Language Processing with Python-Analyzing Text with the Natural Language Toolkit

Steven Bird, Ewan Klein, and Edward Loper

This book explain the implementation of various NLP functionalities like Stemming, removing stop words etc using NLTK Library in python. It has detailed explanations and many illustrated codes to explain each functionality of NLTK Library.

2. Ontology Based Information Extraction- Master of Science Thesis

Carlos Vicient

This work aimed to extract and pre-process data from any kind of Web resource (i.e., plain text and semi-structured documents) in order to generate the required semantically tagged input data for aforementioned semantic data mining techniques. To sum up, in this work it has been designed and implemented a method that is able to extract relevant features from a range of textual documents going from complete plain textual data to semi-structured.

3. Creation and Use of Ontology of Subject Domain "Electrical Engineering"

Le Thanh Tung Nguyen, Matokhina A. V. , Kizim A. V.

In this paper, authors introduce the necessity of creating ontology in "Electrical engineering" region and present it by using OWL language. Protege program is selected as quality tool to design ontology. This ontology can be combined with other region's ontologies in order to create applications, software and specialize products can be used in various areas.

4. Ontology Development 101: A Guide to Creating Your First Ontology

Natalya F. Noy and Deborah L. McGuinness

In this guide, the authors have described an ontology-development methodology for declarative frame-based systems. We listed the steps in the ontology-development process and addressed the complex issues of defining class hierarchies and properties of classes and instances. Ontology design is a creative process and no two ontologies designed by different people would be the same. The potential applications of the ontology and the designer's understanding and view of the domain will undoubtedly affect ontology design choices.

5. Nested Named Entity Recognition

Jenny Rose Finkel and Christopher D. Manning

We presented a discriminative parsing-based method for nested named entity recognition, which does well on both top-level and nested entities. While most NER corpus designers have defenestrated embedded entities, but large amounts of information are lost due to this design decision.

CHAPTER 3
STUDY AREAS AND PROJECT FLOW

3.1 Areas of focus and flow of work

3.1.1 Computer Vision

When you are working with Optical character recognition(OCR) or any data or object recognition problem, the first thing to do is preprocessing. Here preprocessing means to extract the location where our information is located. After extracting the location, any machine algorithm will be performed on that image. This is where Computer Vision (CV) serves its purpose. Computer Vision is the process of using machines to understand and analyze imagery (both photos and videos). Computer Vision is the broad parent name for any computations involving visual content – that means images, videos, icons, and anything else with pixels involved. Here, CV is used for the purpose of box detection.

3.1.2 Optical Character Recognition

Optical character recognition (OCR) refers to both the technology and process of reading and converting typed, printed or handwritten characters into machine-encoded text or something that the computer can manipulate. It is a subset of image recognition and is widely used as a form of data entry with the input being some sort of printed document or data record such as bank statements, sales invoices, passports, resumes and business cards. The document is either scanned or a picture is taken and it is up to the program to recognize the characters and give an output in the form of a text document. Here, we use the OCR library ‘Tesseract’ which can extract text even from low resolution documents

3.1.3 Ontology

Ontology is a structured way of representing the relationship between entities, objects and instances and providing contextual sense when dealing with data in a specialised knowledge domain. Here, ontologies - which provide formal and explicit specifications of conceptualizations - play a crucial role in the information extraction process. Here, we have developed our own ontology for electrical purchase documents which is very useful in providing contextual information in the domain of electrical machines.

3.1.4 Natural Language Processing

Natural Language Processing is the most essential and efficient method to extract and interpret information from textual data using a computer. Though NLP is not exclusively meant for textual data, our implementation revolves around its text based functionalities. Whether it is searching for predefined entities or finding out the part of speech, or to find out the most frequently used word, NLP has it all. These functionalities can be used to extract relationships between entities and other complex features using tools like machine learning. We are using a ML based model to identify named entities and prediction of different variations of a machine description.

3.2 Flow of work

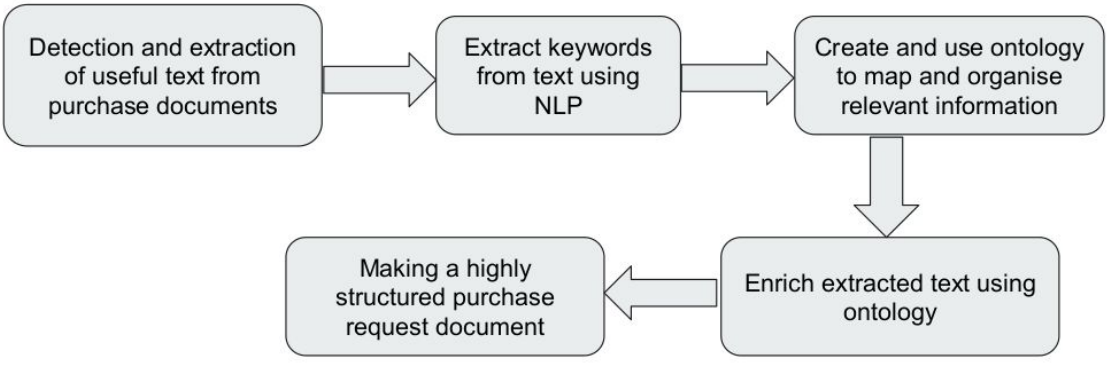


Fig. 3.1 Overall flow of the project

Fig. 3.1 depicts the overall procedure adopted in this project. First, we used CV and OCR to extract raw text. Then, we developed an ontology for electrical products. This was used to train a Named entity recogniser, which can help recognise the relevant entities in the text. Then, SPARQL query was used to extract the named individuals in the ontology, which was compared with the dataset text to create a highly structured output.

CHAPTER 4

ALGORITHMS AND METHODOLOGY

4.1 Box Detection:

It is the primary phase of our project. Here, we adopt the following algorithm to detect the boxes present in the document:

1. Convert the image to grayscale and apply appropriate filters to remove noise.
2. Perform morphological convolution operation of the image with horizontal and vertical kernels to obtain horizontal and vertical lines.
3. Combine horizontal and vertical lines to obtain all the bounding boxes.
4. Find contours (rectangle) and sort them according to size in ascending order.
5. Perform OCR extraction on all the boxes and find the ones having relevant tabular headings.
6. Perform OCR extraction on the boxes right below the tabular column headings in order to extract product information.

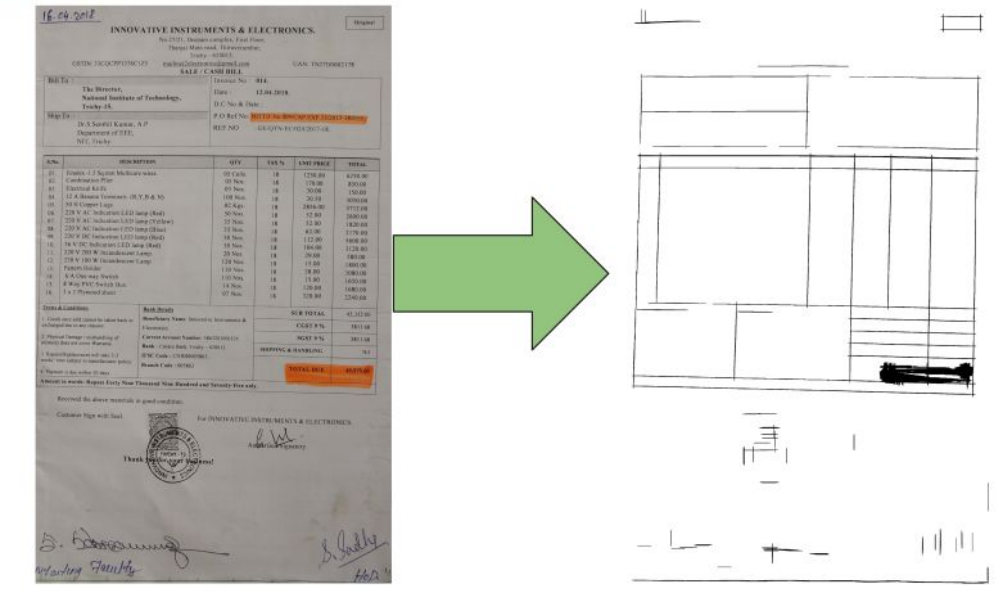


Fig. 4.1 Result of box detection

As depicted in **Fig. 4.1** , box detection was performed using CV in order to find the relevant content present in the document, which can aid in better text extraction.

4.2. Text extraction

After box detection is done, we need to extract text from the appropriate box. We need to find the box that contains product description, and run the Tesseract library API having the selected box as the bounds. For that, we need to find the box containing the word “Description”, and extract text from the box present right beneath it. **Fig. 4.2** gives an appropriate description of the working of the algorithm.

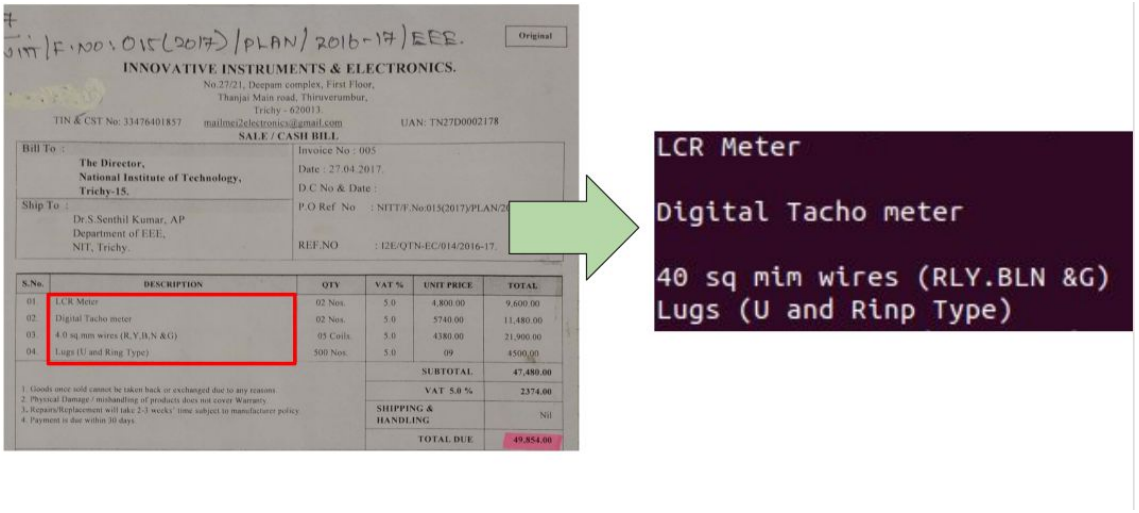


Fig. 4.2. OCR performed on the appropriate section of the document

4.3. Ontology Development

This is the most important phase of the project is the development of ontology. We use Protege, a software which is used for developing ontologies. We use Entity-Relationship (ER) model to represent the ontology. An ER model is used to represent real world objects (Entities) and the relationship between them. Using Protege, we need to create an ontology in the below mentioned fashion:

1. Define Entities (Classes) as shown in Fig 4.3
2. Define Relationships (Object properties) as shown in Fig 4.4
3. Define Attributes for every entity (Data properties) as shown in Fig 4.5

4. Create instances, or individuals for every class (named individuals) as shown in Fig 4.6

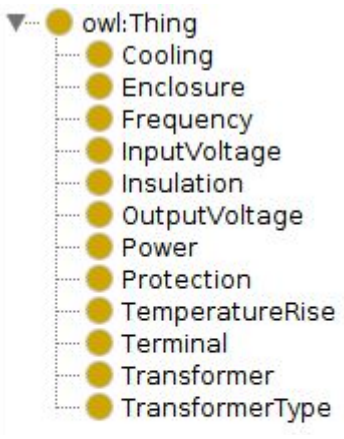


Fig. 4.3. Classes in the ontology

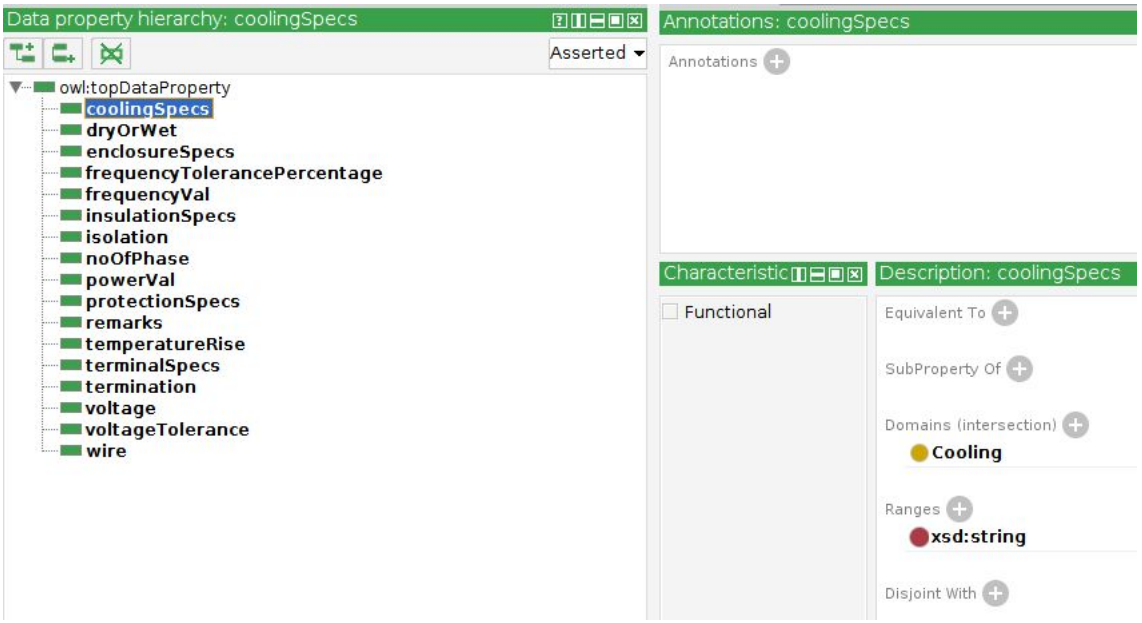


Fig. 4.4. Data properties

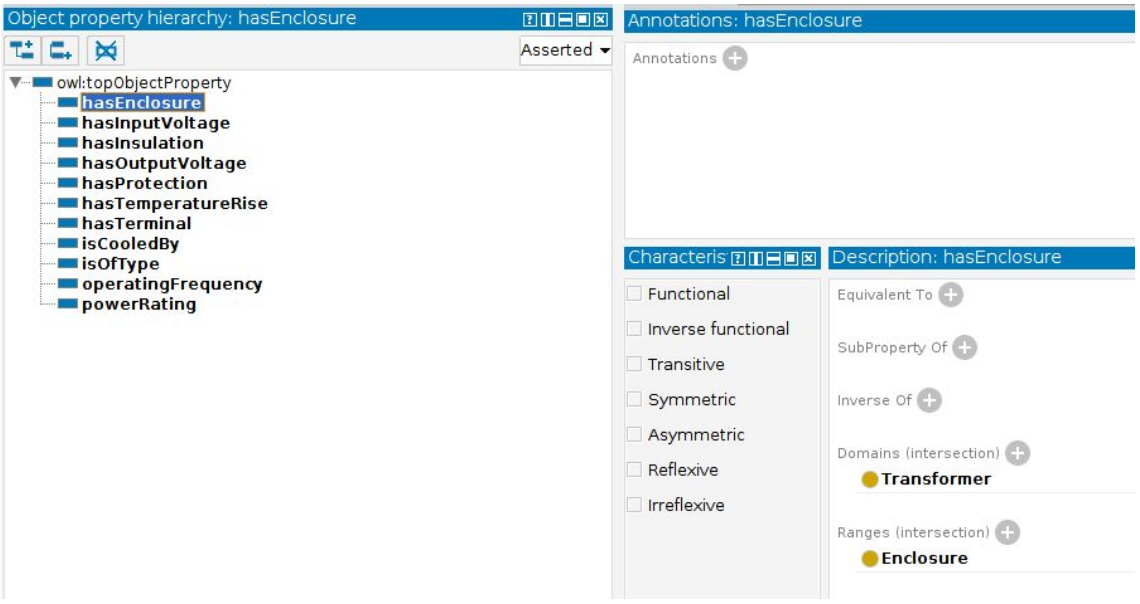


Fig. 4.5. Object properties

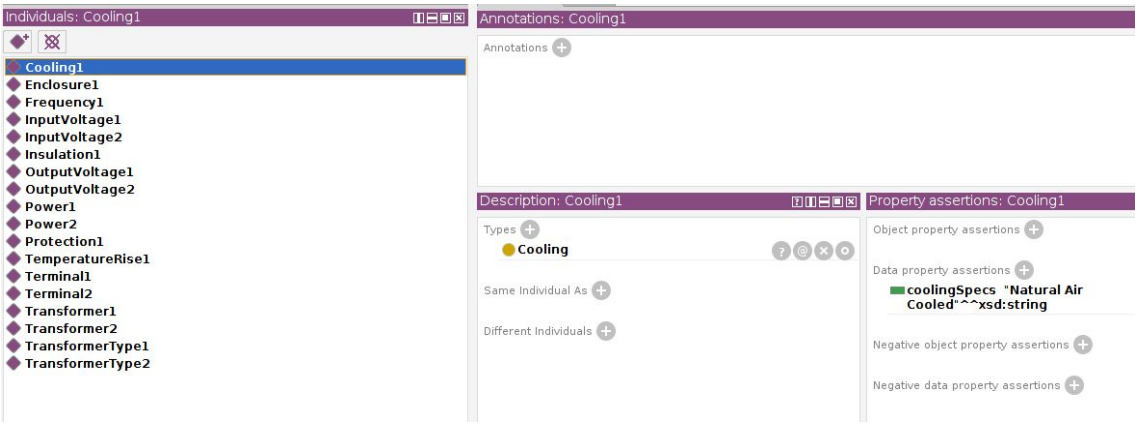


Fig. 4.6. Named individuals

Figure 4.7 is the final state of the ontology developed. Here, the classes are represented as blue circles, and the data properties are represented by blue boxes which connect two circles (classes). Object properties, or attributes of a class is represented as green boxes, and their data type is mentioned in the adjacent yellow boxes.

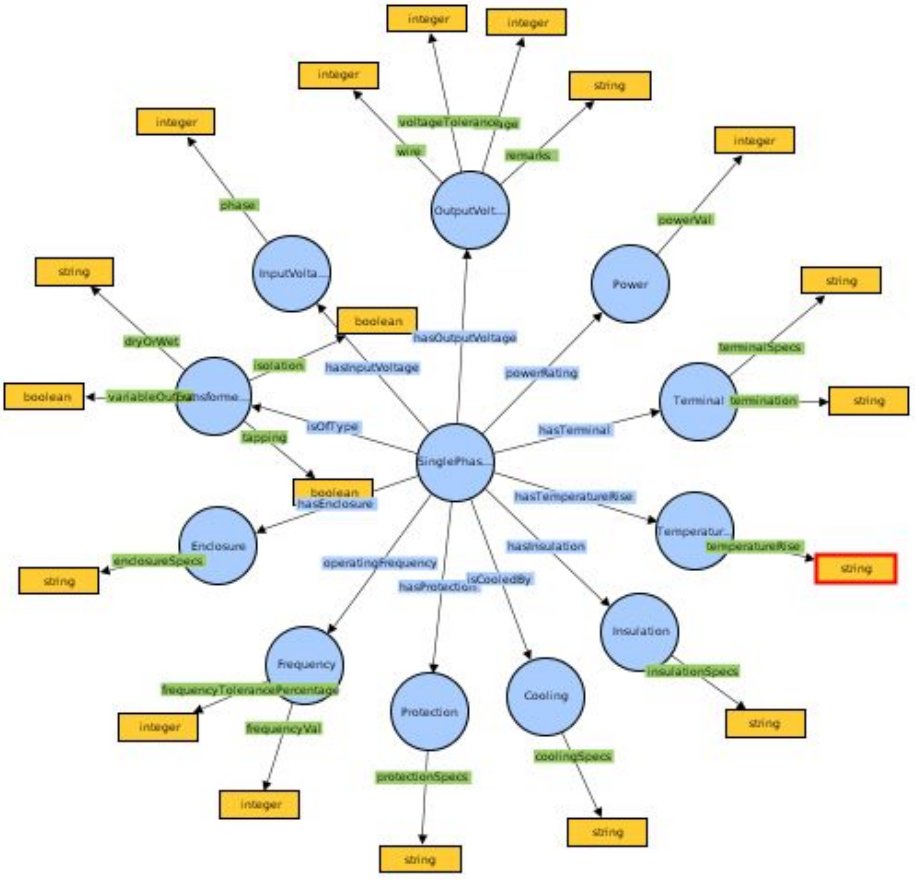


Fig. 4.7. The final result of the ontology

4.4. Natural language processing

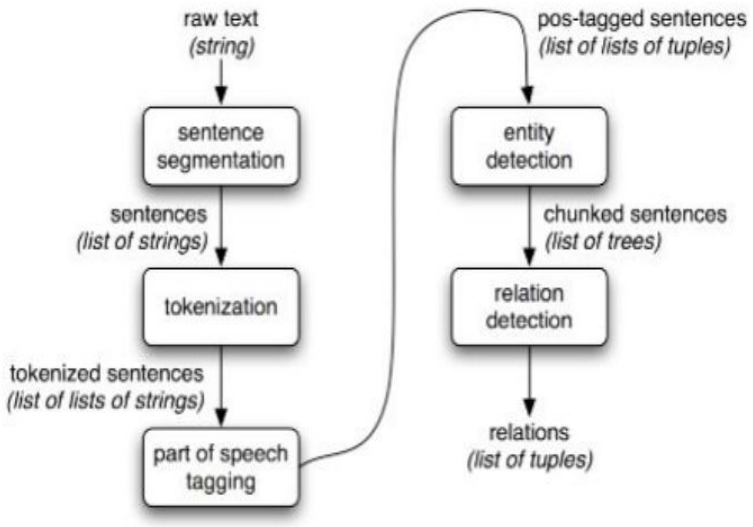


Fig 4.8. Describing various processes involved in Information Extraction.

1- Sentence Segmentation

2- Word Tokenization

3- Parts of Speech Tagging

4- Entity Detection

5- Relation Detection and enhancement of description

Sentence Segmentation

This process is carried out to segment each sentence present in the actual text into a separate entity. This makes a list of all the sentences present in the given text input. This is the initial step and it helps to proceed to the next step. This process is also referred to as sentence tokenizer.

Example:

text = "EEE is Great! I won a lottery."

print(sent_tokenize(text))

Output: ['EEE is Great!', 'I won a lottery ']

Word Tokenization

This process is carried out to segment each word present in the actual text into a separate entity. This makes a list of all the words present in the given text input.

Example:

text = “EEE is Great! I won a lottery."

print(word_tokenize(text))

Output: ['EEE', 'is', 'Great', '!', 'I', 'won', 'a', 'lottery', '.']

Removing the Stop Words

Stop words are words which are filtered out before or after processing of natural language data (text). Though "stop words" usually refers to the most common words in a language, there is no single universal list of stop words used by all natural language processing tools, and indeed not all tools even use such a list. Some tools specifically avoid removing these stop words to support phrase search. This process is very essential as words with very high occurrence frequency is too frequent and words with very low occurrence is too less to impart any change to the actual context. The main advantage is the huge reduction in the memory overhead for the process.

Example:

tokenized_words = ['i', 'am', 'going', 'to', 'go', 'to', 'the', 'store', 'and', 'park']

stop_words = stopwords.words('english')

[word for word in tokenized_words if word not in stop_words]

['going', 'go', 'store', 'park']

Part of Speech Tagging

Parts of speech Tagging is responsible for reading the text in a language and assigning some specific token (Parts of Speech) to each word.

Input: Everything to permit us.

Output: [('Everything', NN),('to', TO), ('permit', VB), ('us', PRP)]

Some important Annotations followed in NLTK library for POS

JJ adjective ‘big’

JJR adjective, comparative ‘bigger’

JJS adjective, superlative ‘biggest’

NN noun, singular ‘desk’

NNS noun plural ‘desks’

NNP proper noun, singular ‘Harrison’

NNPS proper noun, plural ‘Americans’

VB verb, base form take

VBD verb, past tense took

VBG verb, gerund/present participle taking

VBN verb, past participle taken

VBP verb, sing. present, non-3d take

VBZ verb, 3rd person sing. present takes

Named Entity Recognition

This process involves classifying named entities in the text into a particular predefined category. Most of the NLP libraries have functionalities to classify named entities to names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc.

But a highly technical and exclusive application like ours demanded a lot more than the usually used functionalities. Hence we developed a Named Entity Recognizer for Electrical Machines highly inspired in structure and features from the Ontology that we had developed before.

TYPE	DESCRIPTION
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day.
PERCENT	Percentage, including "%".
MONEY	Monetary values, including unit.
QUANTITY	Measurements, as of weight or distance.
ORDINAL	"first", "second", etc.
CARDINAL	Numerals that do not fall under another type.

Fig 4.9. Describing the existing Named Entities in Spacy Library.

SpaCy, which is shown through Fig 4.9 is an open-source software library for advanced Natural Language Processing, written in the programming languages Python and Cython.

So we developed a NER Tagger ourselves.

Processes involved with the development of a new NER Tagger

- Making a full fledged training data set denoting the presence of each entity.
- Creating a blank language class and opening just the ner pipeline. This is mainly because only NER needs to be trained.
- Structuring the model, making the proper annotations and adding entities to the blank language class.
- Training the model several times to improve the results.
- Testing the model with several test data.

We used 4 Named Entities and they are:

- machine
- type(of machine)
- dimension(of machine)
- component(of machine)

A detailed comparison of the NER Tagger we developed along with the Stanford NER Tagger is given in the next section.

Prediction of variations in Machine description

This involved using the Part of Speech Tagger and later using wordnet to find the synonyms for important words with contextual importance. Finally all such extracted words were used to predict the variations that are possible with the description of the Electrical Machine.

CHAPTER 5
RESULTS AND CONCLUSION

Ontology based information extraction

SPARQL query was used to extract the named individuals of the ontology developed. It was used to compare with the OCR extracted text and a structured output was produced as shown in the **Fig. 5.1.**

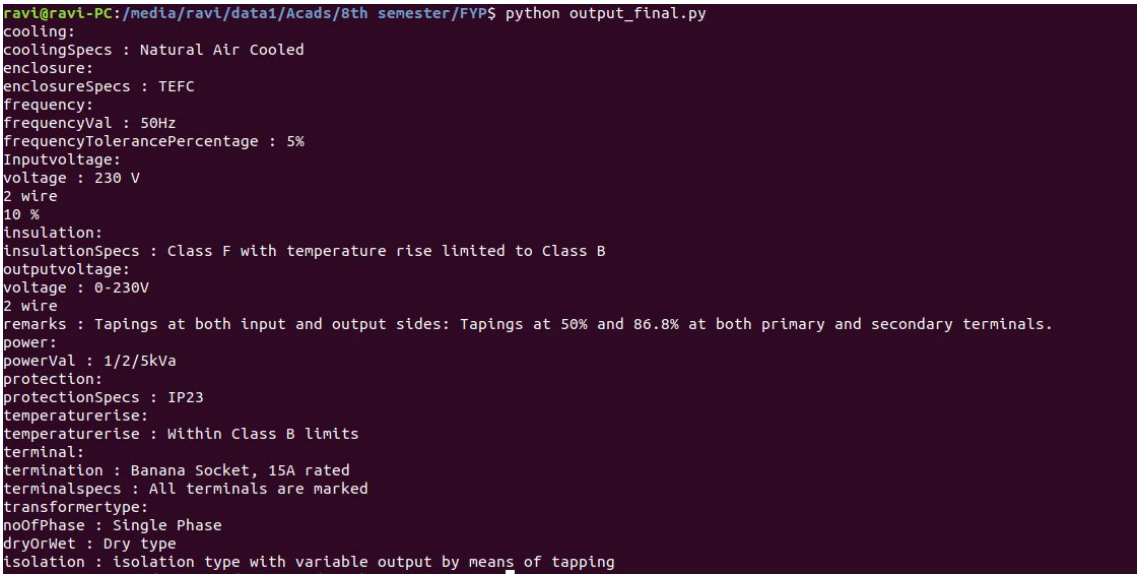


Fig. 5.1. Structured output after extracting information using ontology

Comparison with Stanford NER Tagger

A big benefit of the Stanford NER tagger is that is provides us with a few different models for pulling out named entities. We can use any of the following:

- 3 class model for recognizing locations, persons, and organizations
- 4 class model for recognizing locations, persons, organizations, and miscellaneous entities

- 7 class model for recognizing locations, persons, organizations, times, money, percents, and dates

It is very clear that for our model, none of the classification works. Electrical Machines and data revolving it requires a separate NER Tagger. This is why we developed a NER model using python. A model like this would not work on it's own, it needs to be trained and tested using data to make it more reliable and error free. All these processes were carried out using spaCy library of python. The training data set was restricted by the scarce availability of digital text data on Electrical Machines.

Fig 5.2- Stanford NER Tagger was used to tag an input text.

As shown in the **Fig.5.2** , "o" symbolizes that the word falls out of the 3 predefined entities (PERSON, LOCATION, ORGANIZATION) when NER is carried out in input text.

NER Tagger

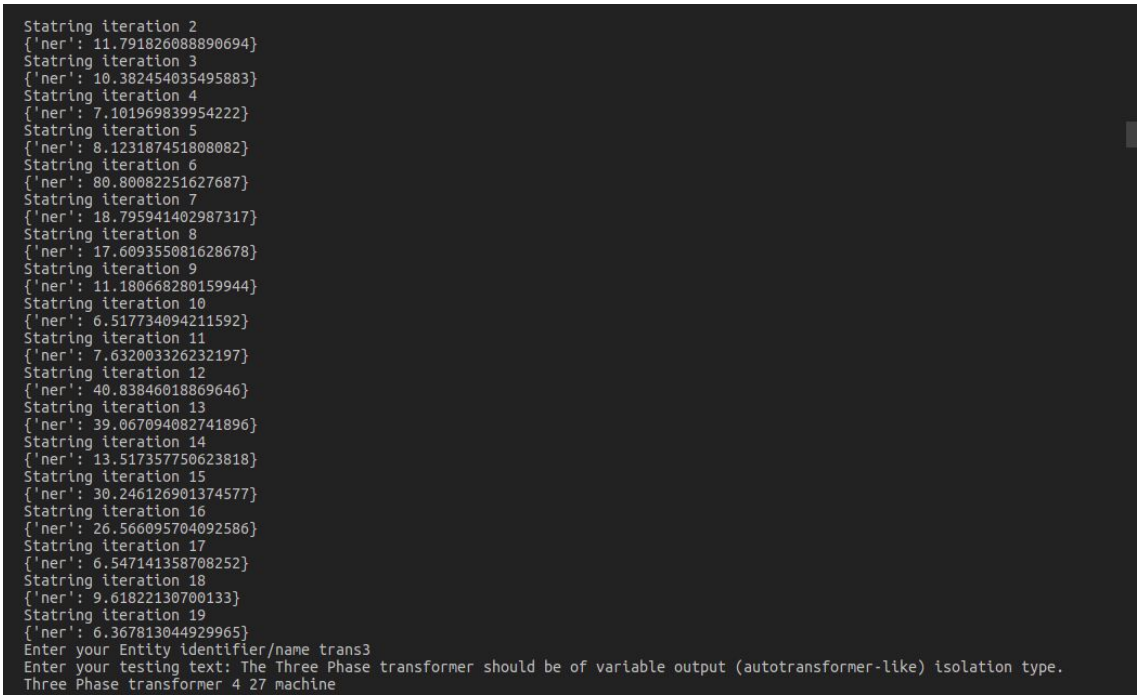


Fig 5.3- 20 Training the model and testing it.

As shown in the Fig 5.3 the result is clearly visible as the “Three Phase transformer” was identified as a “machine”. The named entities predefined in our model are machine, type, dimension and component. And each time an object (word/chunk) is identified as one of amongst from the list, named entity recognition works its way .

Predicting the variations of the Machine description using wordnet

This involved using the Part of Speech Tagger and later using wordnet to find the synonyms for important words with contextual importance. Finally all such extracted words were used to predict the variations that are possible with the description of the Electrical Machine.

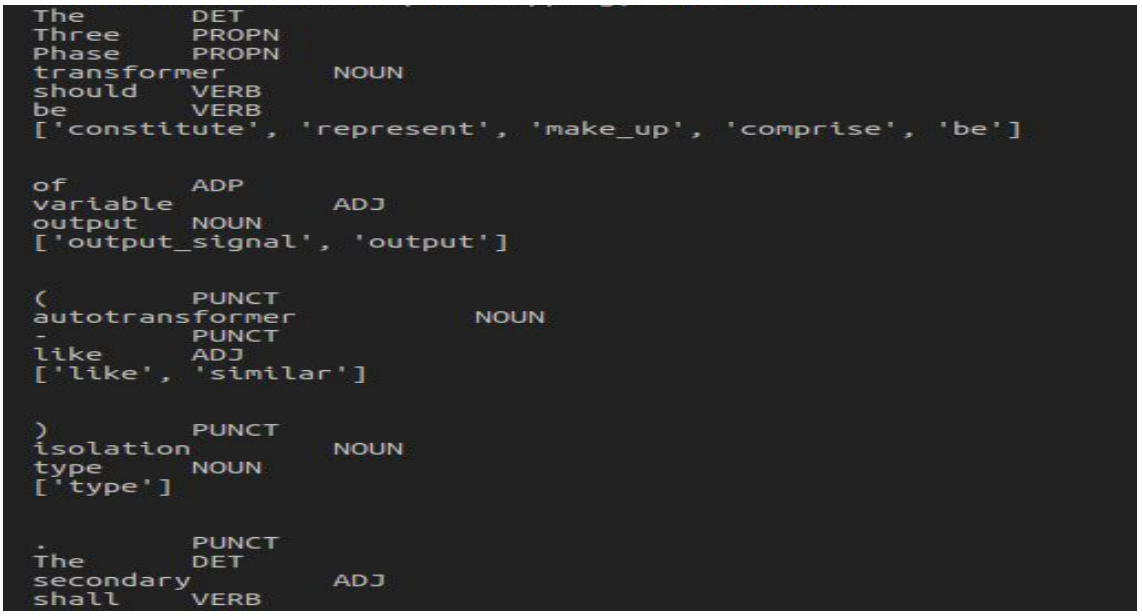


Fig 5.4 -POS Tagging and contextual synonyms

As shown in the Fig 5.4, each token(word) obtained after word tokenization has been associated with its Parts of Speech.

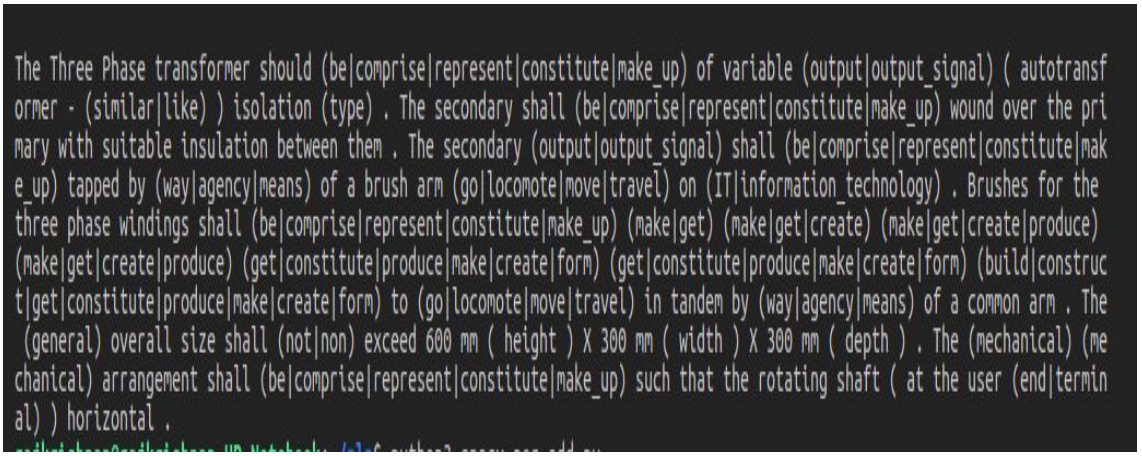


Fig 5.5-Possible variations of the Machine description

Conclusion

An Ontology for Electrical Machines domain was developed using protege. Later Sparql query was carried out to extract named individuals from the ontology in a structured manner. The extracted information was compared with the dataset to create a structured output of the purchase document. SpaCy library of python was used to build a customised NER that could detect four predefined entities from text data. This was trained and tested using domain specific data. Different variations for the Machine description was predicted using wordnet.

Future Prospects

- Extend the Ontology to the entire Electrical Machine domain.
- Extend the NER Tagger with more entities.
- Train the NER model with more data and make it more accurate.
- Prediction of variation in Machine description can be extended for description with a change in structure of the sentences.

REFERENCES

[1] Taehee Lee, Ig-hoon Lee , Suekyung Lee, “Building an operational product ontology system” Electronic Commerce Research and Applications, Volume 5, Issue 1, Spring 2006, Pages 16-28

[2] M.Sc. Stoyanov, Ph.D. Gocheva , Prof. Ph.D. Batchkova, Building an operational product ontology system” Electronic Commerce Research and Applications, Volume 5, Issue 1, Spring 2006, Pages 16-28

[3] Carlos Vicient, Antonio Moreno Ribas, David Sanchez Ruenes, “Ontology-based information extraction”

[4] Bowen Xu, Hui Wang, Huizhong Sun, Yubin Wang, “Design of a Bidirectional Power Converter for Charging Pile based on V2G”, Chinese National Science Foundation [grant number 51577109] and Shandong Provincial Natural Science Foundation [grant number ZR2015EM050], 978-1-5090-9/17/$31.00 c2017IEEE.

[5] Steven Bird, Ewan Klein, and Edward Loper, “Natural Language Processing with Python–Analyzing Text with the Natural Language Toolkit”
2016 IEEE Transportation Electrification Conference band Expo, Asia-Pacific (ITEC), June 1-4, 2016, Busan, Korea, 978-1-5090-1272-5/16/$31.00 c2016 IEEE

[6] K. Rajbabu, Harshavardhan Srinivas, S.Sudha,Industrial information extraction through multi-phase classification using ontology for unstructured documents

APPENDIX

1. CV and OCR extraction code

```
import cv2
import numpy as np
import pytesseract
from math import sqrt

def sort_contours(cnts, method="left-to-right"):
     # initialize the reverse flag and sort index
     reverse = False
     i = 0

     # handle if we need to sort in reverse
     if method == "right-to-left" or method ==
"bottom-to-top":
          reverse = True

     # handle if we are sorting against the y-coordinate
rather than
     # the x-coordinate of the bounding box
     if method == "top-to-bottom" or method ==
"bottom-to-top":
          i = 1

     # construct the list of bounding boxes and sort them from
top to
     # bottom
     boundingBoxes = [cv2.boundingRect(c) for c in cnts]
     (cnts, boundingBoxes) = zip(*sorted(zip(cnts,
boundingBoxes),
          key=lambda b:b[1][i], reverse=reverse))

     # return the list of sorted contours and bounding boxes
     return (cnts, boundingBoxes)

def dist(x1,y1,x2,y2):
     return sqrt((x2-x1)**2 + (y2-y1)**2)
```

```
image=cv2.imread('image.png',0)
(thresh, img_bin) = cv2.threshold(image, 0, 255,
cv2.THRESH_BINARY | cv2.THRESH_OTSU)

img_bin = 255-img_bin

kernel_length = np.array(image).shape[1]//80
verticle_kernel = cv2.getStructuringElement(cv2.MORPH_RECT,
(1, kernel_length))
hori_kernel = cv2.getStructuringElement(cv2.MORPH_RECT,
(kernel_length, 1))
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))

img_temp1 = cv2.erode(img_bin, verticle_kernel, iterations=2)
verticle_lines_img = cv2.dilate(img_temp1, verticle_kernel,
iterations=3)
img_temp2 = cv2.erode(img_bin, hori_kernel, iterations=2)
horizontal_lines_img = cv2.dilate(img_temp2, hori_kernel,
iterations=3)

img_final_bin = cv2.addWeighted(verticle_lines_img, 0.5,
horizontal_lines_img, 0.5, 0.0)
img_final_bin = cv2.erode(~img_final_bin, kernel,
iterations=1)
(thresh, img_final_bin) = cv2.threshold(img_final_bin, 0,255,
cv2.THRESH_BINARY | cv2.THRESH_OTSU)

cv2.imwrite('lines.png',img_final_bin)

im2, contours, hierarchy = cv2.findContours(img_final_bin,
cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
(contours, boundingBoxes) = sort_contours(contours,
method="top-to-bottom")

startx=0
starty=0

for index,c in enumerate(contours):
      x, y, w, h = cv2.boundingRect(c)
      text=pytesseract.image_to_string(image[y:y+h, x:x+w])
```

```
      if 'DESCRIPTION' in text:
            startx=x
            starty=y-h
            break

distance=[]
for c in contours:
      x1, y1, w1, h1 = cv2.boundingRect(c)
      distance.append(dist(x1,y1,startx,starty))

#print distance, min(distance)
description_contour=contours[np.argmin(distance)]
x, y, w, h = cv2.boundingRect(c)
text=pytesseract.image_to_string(image[y:y+h, x:x+w])
print text

cv2.imwrite("output.png",image[y:y+h, x:x+w])
```

2. SPARQL query for ontology extraction

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?subject ?object
      WHERE { ?subject rdfs:subClassOf ?object }
```

3. Training and testing of named entity recogniser

```
import spacy
import random


TRAIN_DATA = [('The Three Phase transformer should be of
variable output (autotransformer-like) isolation type.',
{'entities': [ (4 , 27 , 'machine')  ]}), ('The secondary
shall be wound over the primary with suitable insulation
between them.', {'entities': [(4 , 13 , 'component') ]}),
```

```
('The secondary output shall be tapped by means of a brush arm
moving on it.', {'entities': [(4 , 20 , 'component') ]}),
('Brushes for the three phase windings shall be made to move
in tandem by means of a common arm.', {'entities': [(0 , 7 ,
'component') ]}), ('The general overall size shall not exceed
600 mm (height) X 300 mm (width) X 300 mm (depth).',
{'entities': [(42 , 91 , 'dimension')]}), ('The mechanical
arrangement shall be such that the rotating shaft (at the user
end) horizontal.', {'entities': [(50 , 64 , 'component')]})]


def train_spacy(data,iterations):
    TRAIN_DATA = data
    nlp = spacy.blank('en')  # create blank Language class
    # create the built-in pipeline components and add them to
the pipeline
    # nlp.create_pipe works for built-ins that are registered
with spaCy
    if 'ner' not in nlp.pipe_names:
        ner = nlp.create_pipe('ner')
        nlp.add_pipe(ner, last=True)


    # add labels
    for _, annotations in TRAIN_DATA:
         for ent in annotations.get('entities'):
            ner.add_label(ent[2])

    # get names of other pipes to disable them during training
    other_pipes = [pipe for pipe in nlp.pipe_names if pipe !=
'ner']
    with nlp.disable_pipes(*other_pipes):  # only train NER
        optimizer = nlp.begin_training()
        for itn in range(iterations):
            print("Statring iteration " + str(itn))
            random.shuffle(TRAIN_DATA)
            losses = {}
            for text, annotations in TRAIN_DATA:
                nlp.update(
                    [text],  # batch of texts
```

```
                    [annotations],  # batch of annotations
                    drop=0.2,  # dropout - make it harder to
memorise data
                    sgd=optimizer,  # callable to update
weights
                    losses=losses)
            print(losses)
    return nlp


prdnlp = train_spacy(TRAIN_DATA, 20)

# Save our trained Model
count = 0
while (count < 1):
    count = count + 1
    modelfile = input("Enter your Entity identifier/name ")

    prdnlp.to_disk(modelfile)

    #Test your text
    test_text = input("Enter your testing text: ")
    doc = prdnlp(test_text)
    for ent in doc.ents:
         print(ent.text, ent.start_char, ent.end_char,
ent.label_)
```

4. Prediction of variation in machine description

```
import spacy

from spacy_wordnet.wordnet_annotator import WordnetAnnotator
# Load an spacy model (supported models are "es" and "en")
nlp = spacy.load('en')
nlp.add_pipe(WordnetAnnotator(nlp.lang), after='tagger')
token = nlp('prices')[0]
```

```
# wordnet object link spacy token with nltk wordnet interface
by giving acces to
# synsets and lemmas
token._.wordnet.synsets()
token._.wordnet.lemmas()

# And automatically tags with wordnet domains
token._.wordnet.wordnet_domains()

# Imagine we want to enrich the following sentence with
synonyms
sentence = nlp('The Three Phase transformer should be of
variable output (autotransformer-like) isolation type. The
secondary shall be wound over the primary with suitable
insulation between them. The secondary output shall be tapped
by means of a brush arm moving on it. Brushes for the three
phase windings shall be made to move in tandem by means of a
common arm. The general overall size shall not exceed 600 mm
(height) X 300 mm (width) X 300 mm (depth). The mechanical
arrangement shall be such that the rotating shaft (at the user
end) horizontal.')


# spaCy WordNet lets you find synonyms by domain of interest
# for example economy
engineering_domains = ['engineering', 'technology']
enriched_sentence = []

# For each token in the sentence
for token in sentence:
    # We get those synsets within the desired domains
    synsets =
token._.wordnet.wordnet_synsets_for_domain(engineering_domains
)
    print( token, "\t", token.pos_)
    if synsets:
        lemmas_for_synset = []
        for s in synsets:
            # If we found a synset in the economy domains
            # we get the variants and add them to the enriched
```

```
sentence

            lemmas_for_synset.extend(s.lemma_names())
            print(lemmas_for_synset)
            print("\n")

enriched_sentence.append('({})'.format('|'.join(set(lemmas_for
_synset))))
    else:
        enriched_sentence.append(token.text)


# Let's see our enriched paragraph
print(' '.join(enriched_sentence))
```