

WORD SENSE DISAMBIGUATION : AN INTEGRATED FRAMEWORK

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

MASTER OF SCIENCE

IN

APPLIED MATHEMATICS

Submitted by :

Aditya Sethi

(2K20/MSCMAT/01)

Anjali Aggarwal

(2K20/MSCMAT/04)

Under the supervision of

Dr. Goonjan Jain



DEPARTMENT OF APPLIED MATHEMATICS

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi – 110042

MAY, 2022

M.Sc. APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

We, (Anjali Aggarwal) 2K20/MSCMAT/04, (Aditya Sethi) 2K20/MSCMAT/01 students of M.Sc. (APPLIED MATHEMATICS), hereby declare that the Project Dissertation titled “Word Sense Disambiguation : An Integrated Framework” which is submitted by us to the Department of Applied Mathematics, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Science, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship, or other similar title or recognition.

Place : Delhi

ADITYA SETHI

Date : 5 May 2022

ANJALI AGGARWAL

M.Sc. APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby clarify that the Project Dissertation titled “Word Sense Disambiguation : An Integrated Framework” which is submitted by [Aditya Sethi] 2K20/MSCMAT/01, [Anjali Aggarwal] 2K20/MSCMAT/04 [APPLIED MATHEMATICS], Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Science, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any degree or Diploma to this University or elsewhere.

Place: Delhi

Date: 30-04-2022



Dr. Goonjan Jain

SUPERVISOR

ASSISTANT PROFESSOR
M.Sc. APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

It is not possible to outperform without the assistance & encouragement of respective authorities. This one is certainly no exception.

On the very outset of this report, we would like to extend our sincere & heartfelt obligation towards all the personages who have helped us achieve the completion of this dissertation work. Without their active guidance, help, cooperation & encouragement, we would not have made headway in fulfilment of the results.

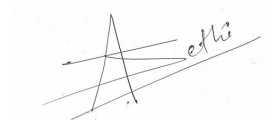
We are ineffably indebted to Dr. Goonjan Jain for her conscientious guidance and encouragement to accomplish our dissertation.

We both are extremely thankful and pay our gratitude towards each other for coordination and maintaining each one's individuality on completion of this project. We extend our gratitude to Delhi Technological University for giving us this opportunity.

We also acknowledge with a deep sense of reverence, our gratitude towards our parents and member of my family, who has always supported us morally as well as economically.

At last but not least gratitude goes to all of our friends who directly or indirectly helped us to complete this project. Any omission in this brief acknowledgement does not mean lack of gratitude.

Thanking You

A handwritten signature in blue ink that reads "Aditya Sethi". The signature is written in a cursive style with a horizontal line underneath the name.

ADITYA SETHI

A handwritten signature in blue ink that reads "Anjali Aggarwal". The signature is written in a cursive style with a horizontal line underneath the name.

ANJALI AGGARWAL

ABSTRACT

Word Sense Disambiguation or WSD intent to find the exact meaning of an ambiguous word for a particular context. Its enormous applications lies in various research areas including sentiment analysis, Information Retrieval, Machine translation and knowledge graph construction. The main objective remains intact that words with same spelling can have completely different senses depending on subtle characteristics of the context. In this paper we analyse the traditional word expert supervised methods. However in comparison with knowledge-based methods, supervised method outperforms better. We standardise the pre-trained BERT and LESK algorithms on SemEval data set and experiment the algorithms on this dataset & compare the accuracy for better results.

TABLE OF CONTENTS

Declaration	ii
Certificate	iii
Acknowledgement	iv
Abstract	v
Table of Contents	vi
Chapter 1 INTRODUCTION	
1.1. Synopsis	1
1.2. Pre-requisite knowledge	2
1.2.1 Supervised WSD approach	2
1.2.2 Knowledge-Based approach	2
Chapter 2 METHODOLOGY	
2.1 Task Definition	3
2.2 Lesk Algorithm	3
2.3 BERT Algorithm	4
2.3.1 BERT (Token-CLS)	4
2.3.2 LSTM NETWORKS	5
2.3.3 Bi-LSTM	5
2.3.4 Gloss BERT	6
2.3.5 Context-Gloss Pairs	6
Chapter 3 EVALUATION	
3.1. Standardising the dataset	7

3.2. Evaluating Lesk algorithm	11
3.3. Evaluating Bert algorithm	12
Chapter 4 INFERENCE	
4.1. Summary	22
4.2. Conclusion	23
4.3. Future Work	23
Appendix : List of Publication	24
References	25

CHAPTER 1 INTRODUCTION

1.1. SYNOPSIS

Word Sense Disambiguation is a primitive challenge in NLP, which intends to find the exact meaning of an ambiguous word in a particular context. The dispute here is that words with the same spelling can have completely different senses or meaning depending on subtle features of the context. A word bear is ambiguous as it can relate either to an animal or to hold on (a verb form).

In this paper we are going to invoke different algorithms used to disambiguate any word in any context and provide integrated framework in identifying the context and exact meaning of the words. Since there is lack of reliable evaluation framework, our comparison of algorithmic results will provide the most appropriately reliable model or framework for Word Sense Disambiguation. We are going to use LESK algorithm which is knowledge based and BERT model for WSD and for comparing the results. We standardised algorithms on SemEval-2015-en data set and experiment the algorithms on this dataset. SemEval-15 task 13 dataset is WSD dataset interpreted with WordNet 3.0. It includes 1022 sense interpretations in four documents taken from three assorted concerns : biomedical, mathematics/computing and social issues.

This papers grants two main features : Firstly a complete evaluation framework for all WSD words overcoming the disadvantages by Standardising the WSD datasets and training dataset semeval-2013-en into unified format. Initially cleaning the dataset and converting the ambiguous word from dataset to WordNet 3.0. Secondly, we evaluate this framework to perform a fair comparison of proposed techniques available in WSD literature.

1.2 PRE-REQUISITE KNOWLEDGE

The main aim of Word Sense Disambiguation is to link the words in subject to the most appropriate meaning in a pre-defined sense inventory. In this paper, WSD set-up is segregated into two major groups : Supervised and knowledge- based.

The following is summarisation of these types of approach :

1.2.1. SUPERVISED WSD APPROACH :

Supervised approach is based on a training corpus of words tagged with their respective sense. This approach is mostly based on facts provided by words neighbouring the targeted word and its collocations. Supervised models have conventionally outperformed knowledge-based approach. However, acquiring sense-annotated corpora is quite upscale, and sometimes such compilations are not even available for specific concerns. This is the reason for the supervised methods to count on unlabelled corpora.

1.2.2. KNOWLEDGE-BASED APPROACH :

Knowledge- based approach does not require any sense-annotated corpus. Rather , this approach depends on the context of frequently updated knowledge resources for disambiguation. The one of major approaches is Lesk algorithm introduced in 1986 in which its initial version consisted of calculating the extend over the context of the target word and its definition as mentioned in sense inventory.

CHAPTER 2 METHODOLOGY

2.1. TASK DEFINITION :

We have used Semeval-2015-en dataset for its implementation on proposed algorithms. Data is in form of a xml file that has various sentences that contains ambiguous words. Our first task is to clean the dataset and then with the help of already present nltk corpus and wordnet, we disambiguate the searched word from the dataset Semeval-2015-en , and scan for its synonyms and antonyms with their first senses present in annotated corpus. Then our task will be to implement the clean data achieved to proposed Lesk and Bert algorithm.

2.2. LESK ALGORITHM :

Lesk algorithm tells us that the accurate sense of words individually in a given context which is identified by detecting the meaning that overlaps between the dictionary definition and the given context. Instead of identifying the meanings of all words at the same time in a given context, this approach addresses each word individually, irrespective of the meaning of the other words occurring in the same context.

The Lesk-algorithm scans sense of words in short clauses. The definition of each sense of a word in a clause is collated to the meaning of every other word in the sentence. A word is allocated the meaning whose gloss sense contributes to the maximum number of common words with the glosses of the other words. This algorithm starts a new process for each word and does not utilize the senses it previously assigned.

BERT ALGORITHM :

BERT is a characterization model, and it is a multilayer bidirectional Transformer encoder. BERT model is pre-trained on a large collection of written text and two unsupervised prediction pieces of work, that is the next sentence prediction tasks and masked language model are used in pre-training. When we use BERT for WSD tasks, the tuning up procedure is recommended. The pretrained BERT model on WSD task is tuned up.

2.3.1. BERT (Token-CLS) :

Neural networks are the web of interrelated vertices where each vertex performs simple calculations. To get appropriate results we take union of calculations. In present machine learning and deep learning environment, neural networks are considered among the most important area of study growing in readiness. [3] Three major neural networks are ANN, CNN and RNN.

2.3.2. LSTM Networks :

Long Short-Term Memory networks are influential kind of RNN. LSTM were introduced to curb the long-term dependency problem. In regular RNN, the

frequent problem is occurred when we connect new information to previous information. If RNN is able to do this, the information would be very useful. This problem is called long-term dependency.

2.3.3. Bi-LSTM :

Bidirectional Long-Short Term Memory is the procedure of making any neural network to collect the sequence information in both directions that is future to past or past to future. We need to scan each target word in the mentioned sentence to find for disambiguation. Basically, WSD task is a token level classification task. To embrace BERT to WSD task, we take the final concealed state of the token correlating to the target word, if there is more than one token, we take their average and add a classification layer for every target word, which is similar to the last layer of the Bi-LSTM model.

2.3.4. GlossBert :

BERT is favourable to model the relationship of pair of texts for various NLP tasks. Now in order to collect the complete facts of the target word in WordNet, GlossBERT is used to set up context-gloss pairs. Hence it treats WSD tasks as a classification problem for sentence-pair.

2.3.5 Context-Gloss Pairs :

The sentence which has target words is referred as *context* sentence. We extract glosses of all N possible senses of the target word in WordNet To obtain the

gloss sentence for each target word. We add [CLS] and [SEP] tokens to the *contextgloss* pairs for making it satisfactory for the input of BERT model.

CHAPTER 3 EVALUATION

The evaluation scheme consists of the WSD evaluation datasets described in Section 2.2 and 2.3. To proceed with evaluation we use mentioned scheme to implement assorted WSD set-up. The systems used in the evaluation are described in detail in Chapter 2, the results are shown in Section 4.

3.1. STANDARDISING THE WSD DATASET SEMEVAL-2013-EN INTO UNIFIED FORMAT :

First 5 sentences of the dataset are enclosed under tags `<sentence>` `</sentence>` as shown below

SENTENCE 1

```
<corpus lang="en">
<text id="d001">
<sentence id="d001.s001">
<wf id="d001.s001.t001" pos="X">This</wf>
<wf id="d001.s001.t002" lemma="document" pos="N">document</wf>
<wf id="d001.s001.t003" lemma="be" pos="V">is</wf>
<wf id="d001.s001.t004" pos="X">a</wf>
<wf id="d001.s001.t005" lemma="summary" pos="N">summary</wf>
<wf id="d001.s001.t006" pos="X">of</wf>
<wf id="d001.s001.t007" pos="X">the</wf>
<wf id="d001.s001.t008" lemma="european" pos="J">European</wf>
<wf id="d001.s001.t009" lemma="public" pos="J">Public</wf>
<wf id="d001.s001.t010" lemma="assessment" pos="N">Assessment</wf>
<wf id="d001.s001.t011" lemma="report" pos="N">Report</wf>
<wf id="d001.s001.t012" pos="X">(</wf>
<wf id="d001.s001.t013" lemma="epar" pos="N">EPAR</wf>
<wf id="d001.s001.t014" pos="X">)</wf>
<wf id="d001.s001.t015" pos="X">.</wf>
</sentence>
```

SENTENCE 2

```

<sentence id="d001.s002">
<wf id="d001.s002.t001" pos="X">@It</wf>
<wf id="d001.s002.t002" lemma="explain" pos="V">explains</wf>
<wf id="d001.s002.t003" pos="X">how</wf>
<wf id="d001.s002.t004" pos="X">the</wf>
<wf id="d001.s002.t005" lemma="committee" pos="N">Committee</wf>
<wf id="d001.s002.t006" pos="X">for</wf>
<wf id="d001.s002.t007" lemma="medicinal" pos="J">Medicinal</wf>
<wf id="d001.s002.t008" lemma="product" pos="N">Products</wf>
<wf id="d001.s002.t009" pos="X">for</wf>
<wf id="d001.s002.t010" lemma="human" pos="J">Human</wf>
<wf id="d001.s002.t011" lemma="use" pos="N">Use</wf>
<wf id="d001.s002.t012" pos="X">(</wf>
<wf id="d001.s002.t013" lemma="chmp" pos="N">CHMP</wf>
<wf id="d001.s002.t014" pos="X">)</wf>
<wf id="d001.s002.t015" lemma="assess" pos="V">assessed</wf>
<wf id="d001.s002.t016" pos="X">the</wf>
<wf id="d001.s002.t017" lemma="study" pos="N">studies</wf>
<wf id="d001.s002.t018" lemma="perform" pos="V">performed</wf>
<wf id="d001.s002.t019" pos="X">,</wf>
<wf id="d001.s002.t020" pos="X">to</wf>
<wf id="d001.s002.t021" lemma="reach" pos="V">reach</wf>
<wf id="d001.s002.t022" pos="X">their</wf>
<wf id="d001.s002.t023" lemma="recommendation" pos="N">recommendations</wf>
<wf id="d001.s002.t024" pos="X">on</wf>
<wf id="d001.s002.t025" pos="X">how</wf>
<wf id="d001.s002.t026" pos="X">to</wf>
<wf id="d001.s002.t027" lemma="use" pos="V">use</wf>
<wf id="d001.s002.t028" pos="X">the</wf>
<wf id="d001.s002.t029" lemma="medicine" pos="N">medicine</wf>
<wf id="d001.s002.t030" pos="X">.</wf>
</sentence>

```

SENTENCE 3

```

<sentence id="d001.s003">
<wf id="d001.s003.t001" pos="X">@If</wf>
<wf id="d001.s003.t002" pos="X">you</wf>
<wf id="d001.s003.t003" lemma="need" pos="V">need</wf>
<wf id="d001.s003.t004" lemma="more" pos="J">more</wf>
<wf id="d001.s003.t005" lemma="information" pos="N">information</wf>
<wf id="d001.s003.t006" pos="X">about</wf>
<wf id="d001.s003.t007" pos="X">your</wf>
<wf id="d001.s003.t008" lemma="medical" pos="J">medical</wf>
<wf id="d001.s003.t009" lemma="condition" pos="N">condition</wf>
<wf id="d001.s003.t010" pos="X">or</wf>
<wf id="d001.s003.t011" pos="X">your</wf>
<wf id="d001.s003.t012" lemma="treatment" pos="N">treatment</wf>
<wf id="d001.s003.t013" pos="X">,</wf>
<wf id="d001.s003.t014" lemma="read" pos="V">read</wf>
<wf id="d001.s003.t015" pos="X">the</wf>
<wf id="d001.s003.t016" lemma="package" pos="N">Package</wf>
<wf id="d001.s003.t017" lemma="leaflet" pos="N">Leaflet</wf>
<wf id="d001.s003.t018" pos="X">(</wf>
<wf id="d001.s003.t019" lemma="also" pos="R">also</wf>
<wf id="d001.s003.t020" lemma="part" pos="N">part</wf>
<wf id="d001.s003.t021" pos="X">of</wf>
<wf id="d001.s003.t022" pos="X">the</wf>
<wf id="d001.s003.t023" lemma="epar" pos="N">EPAR</wf>
<wf id="d001.s003.t024" pos="X">)</wf>
<wf id="d001.s003.t025" pos="X">or</wf>
<wf id="d001.s003.t026" lemma="contact" pos="V">contact</wf>
<wf id="d001.s003.t027" pos="X">your</wf>
<wf id="d001.s003.t028" lemma="doctor" pos="N">doctor</wf>
<wf id="d001.s003.t029" pos="X">or</wf>
<wf id="d001.s003.t030" lemma="pharmacist" pos="N">pharmacist</wf>
<wf id="d001.s003.t031" pos="X">.</wf>
</sentence>

```

SENTENCE 4

```

<sentence id="d001.s004">
<wf id="d001.s004.t001" pos="X">@If</wf>
<wf id="d001.s004.t002" pos="X">you</wf>
<wf id="d001.s004.t003" lemma="want" pos="V">want</wf>
<wf id="d001.s004.t004" lemma="more" pos="J">more</wf>
<wf id="d001.s004.t005" lemma="information" pos="N">information</wf>
<wf id="d001.s004.t006" pos="X">on</wf>
<wf id="d001.s004.t007" pos="X">the</wf>
<wf id="d001.s004.t008" lemma="basis" pos="N">basis</wf>
<wf id="d001.s004.t009" pos="X">of</wf>
<wf id="d001.s004.t010" pos="X">the</wf>
<wf id="d001.s004.t011" lemma="chmp" pos="N">CHMP</wf>
<wf id="d001.s004.t012" lemma="recommendation" pos="N">recommendations</wf>
<wf id="d001.s004.t013" pos="X">,</wf>
<wf id="d001.s004.t014" lemma="read" pos="V">read</wf>
<wf id="d001.s004.t015" pos="X">the</wf>
<wf id="d001.s004.t016" lemma="scientific" pos="J">Scientific</wf>
<wf id="d001.s004.t017" lemma="discussion" pos="N">Discussion</wf>
<wf id="d001.s004.t018" pos="X">(</wf>
<wf id="d001.s004.t019" lemma="also" pos="R">also</wf>
<wf id="d001.s004.t020" lemma="part" pos="N">part</wf>
<wf id="d001.s004.t021" pos="X">of</wf>
<wf id="d001.s004.t022" pos="X">the</wf>
<wf id="d001.s004.t023" lemma="epar" pos="N">EPAR</wf>
<wf id="d001.s004.t024" pos="X">)</wf>
<wf id="d001.s004.t025" pos="X">.</wf>
</sentence>

```

SENTENCE 5

```

<sentence id="d001.s005">
<wf id="d001.s005.t001" pos="X">@What</wf>
<wf id="d001.s005.t002" lemma="be" pos="V">is</wf>
<wf id="d001.s005.t003" lemma="alimta" pos="N">Alimta</wf>
<wf id="d001.s005.t004" pos="X">?</wf>
</sentence>

```

After standardising the dataset, first 5 sentences will be :

' This document is a summary of the European Public Assessment Report (EPAR) . '

'It explains how the Committee for Medicinal Products for Human Use (CHMP) assessed the studies performed , to reach their recommendations on how to use the medicine . '

'If you need more information about your medical condition or your treatment , read the Package Leaflet (also part of the EPAR) or contact your doctor or pharmacist . '

'If you want more information on the basis of the CHMP recommendations , read the Scientific Discussion (also part of the EPAR) . '

'What is Alimta ? '

Now for instance let's suppose that ambiguous word in this sentence is 'leaflet', then its synset meaning , synonym and first few senses present in wordnet are :

Enter a word for its disambiguation in file : leaflet

Word found!!

Synset meaning : a thin triangular flap of a heart valve

Synonym for leaflet is: {'brochure', 'pamphlet', 'booklet', 'cusp', 'folder', 'leaflet'}

<bound method Synset.name of Synset('cusp.n.02')> : a thin triangular flap of a heart valve

Synset('cusp.n.02') : a thin triangular flap of a heart valve

Synset('leaflet.n.02') : part of a compound leaf

Synset('booklet.n.01') : a small book usually having a paper cover

3.2. LESK ALGORITHM :

[5] SemEval-15 task 13 is the WSD dataset interpreted with WordNet 3.0. It consists of 1022 sense annotations. In less algorithm, the correct sense for every word in a context is obtained independently by tracking the sense that superimposes the most among its dictionary definition and the provided context.

PYTHON CODE

```

from nltk.corpus import wordnet as wn
from nltk.stem import PorterStemmer
from itertools import chain

ps = PorterStemmer()

def lesk(context_sentence,ambiguous_word, pos=None, stem=True, hyperhypo=True):
    max_overlaps = 0; lesk_sense = None
    context_sentence=context_sentence.split()
    for ss in wn.synsets(ambiguous_word):
        # If POS is specified.
        if pos and ss.pos is not pos:
            continue

        lesk_dictionary = []

        # Includes definition.
        lesk_dictionary=lesk_dictionary+ss.definition().split()
        # Includes lemma_names.
        lesk_dictionary=lesk_dictionary+ss.lemma_names()

        # Optional: includes lemma_names of hypernyms and hyponyms.
        if hyperhypo == True:
            lesk_dictionary+= list(chain(*[i.lemma_names() for i in ss.hypernyms()+ss.hyponyms()]))

        if stem == True: # Matching exact words causes sparsity, so lets match stems.
            lesk_dictionary = [ps.stem(i) for i in lesk_dictionary]
            context_sentence = [ps.stem(i) for i in context_sentence]

        overlaps = set(lesk_dictionary).intersection(context_sentence)

        if len(overlaps) > max_overlaps:
            lesk_sense = ss
            max_overlaps = len(overlaps)
    return lesk_sense

print("Context:", list_of_sentences[15])
w=input('Enter the word for its disambiguation :')
answer = lesk(list_of_sentences[15],w)
print("\nSense:", answer)
print("\nDefinition:",answer.definition())

```

In any sentence for chosen target word, its synset definition is scanned,

For a random sentence in dataset mentioned :

Context: It is given once every three weeks as an intravenous infusion lasting 10 minutes .
Enter the word for its disambiguation :intravenous

Sense: Synset('intravenous.a.01')

Definition: within or by means of a vein

3.3. BERT ALGORITHM :

We fine-tune a BERT model using the `pytorch_transformers-models` PIP package. [3] To do this, we use the pre-trained BERT encoder (large-uncased BERT) from `pytorch_transformers` Hub. To fine-tune a pre-trained model, exactly the same tokenization, vocabulary, and index mapping with the model should be used. So, we use the tokenizer that was used by the base model.

Next, we initialize the provided data and prepare it as the input of the BERT model. For every sentence, we add a “[TGT]” token before and after the target word in each sentence. After that, we add a “[SEP]” (Separator) token at the end of each sentence. Then we encrypt them separately by the tokenizer. We also encode and add a “[CLS]” token at the first position to be able to do a classification task. Note that we had to add “[TGT]” as a token into the tokenizer vocabulary. Here is an example with “[TGT]” token.

```

import os
import zipfile

bert_wsd_pytorch = "/content/gdrive/My Drive/bert_base-augmented-
batch_size=128-lr=2e-5-max_gloss=6.zip"
extract_directory = "/content/gdrive/My Drive"

extracted_folder = bert_wsd_pytorch.replace(".zip", "")

# If unzipped folder exists don't unzip again.
if not os.path.isdir(extracted_folder):
    with zipfile.ZipFile(bert_wsd_pytorch, 'r') as zip_ref:
        zip_ref.extractall(extract_directory)
else:
    print (extracted_folder, " is extracted already")

```

```

import torch
import math
from pytorch_transformers import BertModel, BertConfig, BertPreTrainedModel,
BertTokenizer

class BertWSD(BertPreTrainedModel):
    def __init__(self, config):
        super().__init__(config)

        self.bert = BertModel(config)
        self.dropout = torch.nn.Dropout(config.hidden_dropout_prob)

        self.ranking_linear = torch.nn.Linear(config.hidden_size, 1)

        self.init_weights()

    # def _forward(args, model, batch):
    #     batch = tuple(t.to(args.device) for t in batch)
    #     outputs = model.bert(input_ids=batch[0], attention_mask=batch[1], token_type_ids=batch[2])

    #     return model.dropout(outputs[1])

DEVICE = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model_dir = "/content/gdrive/My Drive/bert_base-augmented-batch_size=128-
lr=2e-5-max_gloss=6"

model = BertWSD.from_pretrained(model_dir)
tokenizer = BertTokenizer.from_pretrained(model_dir)
# add new special token
if '[TGT]' not in tokenizer.additional_special_tokens:
    tokenizer.add_special_tokens({'additional_special_tokens': ['[TGT]']})
    assert '[TGT]' in tokenizer.additional_special_tokens
    model.resize_token_embeddings(len(tokenizer))

model.to(DEVICE)
model.eval()

```

```

import csv
import os
from collections import namedtuple

import nltk
nltk.download('wordnet')
from nltk.corpus import wordnet as wn

import torch
from tqdm import tqdm

GlossSelectionRecord = namedtuple("GlossSelectionRecord",
                                  ["guid", "sentence",
                                   "sense_keys", "glosses", "targets"])
BertInput = namedtuple("BertInput",
                      ["input_ids", "input_mask",
                       "segment_ids", "label_id"])

def _create_features_from_records(records, max_seq_length, tokenizer,
                                 cls_token_at_end=False, pad_on_left=False,
                                 cls_token='[CLS]', sep_token='[SEP]',
                                 pad_token=0,
                                 sequence_a_segment_id=0,
                                 sequence_b_segment_id=1,
                                 cls_token_segment_id=1,
                                 pad_token_segment_id=0,
                                 mask_padding_with_zero=True,
                                 disable_progress_bar=False):
    """ Convert records to list of features.
    Each feature is a list of sub-features where the first element is
    always the feature created from context-gloss pair while
    the rest of the elements are features created from
    context-example pairs (if available)
    `cls_token_at_end` define the location of the CLS token:
    - False (Default, BERT/XLM pattern): [CLS] + A + [SEP] + B + [SEP]
    - True (XLNet/GPT pattern): A + [SEP] + B + [SEP] + [CLS]
    `cls_token_segment_id` define
    the segment id associated to the CLS token (0 for BERT, 2 for XLNet)
    """

```

```

features = []
for record in tqdm(records, disable=disable_progress_bar):
    tokens_a = tokenizer.tokenize(record.sentence)

    sequences = [(gloss, 1 if i in record.targets else 0)
                 for i, gloss in enumerate(record.glosses)]

    pairs = []
    for seq, label in sequences:
        tokens_b = tokenizer.tokenize(seq)

        # Modifies `tokens_a` and `tokens_b`
        # in place so that the total
        # length is less than the specified length.
        # Account for [CLS], [SEP], [SEP] with "- 3"
        _truncate_seq_pair(tokens_a, tokens_b, max_seq_length - 3)

```

```

tokens = tokens_a + [sep_token]
segment_ids = [sequence_a_segment_id] * len(tokens)

tokens += tokens_b + [sep_token]
segment_ids += [sequence_b_segment_id] * (len(tokens_b) + 1)

if cls_token_at_end:
    tokens = tokens + [cls_token]
    segment_ids = segment_ids + [cls_token_segment_id]
else:
    tokens = [cls_token] + tokens
    segment_ids = [cls_token_segment_id] + segment_ids

input_ids = tokenizer.convert_tokens_to_ids(tokens)

# The mask has 1 for real tokens and 0 for padding tokens.
# Only real tokens are attended to.
input_mask = [1 if mask_padding_with_zero else 0] * len(input_ids)

# Zero-pad up to the sequence length.
padding_length = max_seq_length - len(input_ids)
if pad_on_left:
    input_ids = ([pad_token] * padding_length) + input_ids
    input_mask = ([0 if mask_padding_with_zero else 1]
                  * padding_length) + input_mask
    segment_ids = ([pad_token_segment_id]
                   * padding_length) + segment_ids
else:
    input_ids = input_ids + ([pad_token] * padding_length)
    input_mask = input_mask + ([0 if mask_padding_with_zero else 1]
                                * padding_length)
    segment_ids = segment_ids + ([pad_token_segment_id]
                                  * padding_length)

assert len(input_ids) == max_seq_length
assert len(input_mask) == max_seq_length
assert len(segment_ids) == max_seq_length

pairs.append(
    BertInput(input_ids=input_ids, input_mask=input_mask,
              segment_ids=segment_ids, label_id=label)
)

features.append(pairs)

return features

```

```

def _truncate_seq_pair(tokens_a, tokens_b, max_length):
    """Truncates a sequence pair in place to the maximum length."""

    # This is a simple heuristic which will always truncate the longer sequence
    # one token at a time. This makes more sense than truncating an equal percent
    # of tokens from each, since if one sequence is very short then each token
    # that's truncated likely contains more information than a longer sequence.
    while True:
        total_length = len(tokens_a) + len(tokens_b)
        if total_length <= max_length:
            break
        if len(tokens_a) > len(tokens_b):
            tokens_a.pop()
        else:
            tokens_b.pop()

```

```

import re

import torch
from tabulate import tabulate
from torch.nn.functional import softmax
from tqdm import tqdm
from pytorch_transformers import BertTokenizer

import time

from transformers import T5ForConditionalGeneration, T5Tokenizer

question_model = T5ForConditionalGeneration.from_pretrained(
    'ramsrigouthamg/t5_squad_v1')
question_tokenizer = T5Tokenizer.from_pretrained('t5-base')

MAX_SEQ_LENGTH = 128

```

```

def get_sense(sent):

    re_result = re.search(r"\[TGT\] (.*)\[TGT\]", sent)
    if re_result is None:
        print("\nIncorrect input format. Please try again.")

    ambiguous_word = re_result.group(1).strip()

    results = dict()

    wn_pos = wn.NOUN
    for i, synset in enumerate(set(wn.synsets(ambiguous_word,
                                             pos=wn_pos))):
        results[synset] = synset.definition()

    if len(results) == 0:
        return (None, None, ambiguous_word)

```

```

# Distractors from Wordnet
def get_distractors_wordnet(syn, word):
    distractors=[]
    word= word.lower()
    orig_word = word
    if len(word.split())>0:
        word = word.replace(" ", "_")
    hypernym = syn.hypernyms()
    if len(hypernym) == 0:
        return distractors
    for item in hypernym[0].hyponyms():
        name = item.lemmas()[0].name()
        #print ("name ",name, " word",orig_word)
        if name == orig_word:
            continue
        name = name.replace(" ", " ")
        name = " ".join(w.capitalize() for w in name.split())
        if name is not None and name not in distractors:
            distractors.append(name)
    return distractors

```

```

def get_question(sentence, answer):
    text = "context: {} answer: {} </s>".format(sentence, answer)
    max_len = 256
    encoding = question_tokenizer.encode_plus(text, max_length=max_len,
                                              pad_to_max_length=True,
                                              return_tensors="pt")

    input_ids, attention_mask = encoding["input_ids"], encoding["attention_mask"]

    outs = question_model.generate(input_ids=input_ids,
                                  attention_mask=attention_mask,
                                  early_stopping=True,
                                  num_beams=5,
                                  num_return_sequences=1,
                                  no_repeat_ngram_size=2,
                                  max_length=200)

    dec = [question_tokenizer.decode(ids) for ids in outs]

    Question = dec[0].replace("question:", "")
    Question= Question.strip()
    return Question

```

```

def disambiguate(sent):
    sentence_for_bert = sent.replace("**", " [TGT] ")
    sentence_for_bert = " ".join(sentence_for_bert.split())
    # try:
    sense, meaning, answer = get_sense(sentence_for_bert)
    if sense is not None:
        distractors = get_distractors_wordnet(sense, answer)
    else:
        distractors = ["Word not found in Wordnet. So unable to extract distractor
s."]
    sentence_for_T5 = sent.replace("**", " ")
    sentence_for_T5 = " ".join(sentence_for_T5.split())
    ques = get_question(sentence_for_T5, answer)
    return ques, answer, distractors, meaning

sentence = ("This document is a **summary** of the European Public Assessment
Report ( EPAR )")

question, answer, distractors, meaning = disambiguate(sentence)

print (question)
print (answer)
print (distractors)
print (meaning)

```

What type of document is the European Public Assessment Report?</s>

summary

['Agreement', 'Amendment', 'Announcement', 'Answer', 'Assurance', 'Bid', 'Bill Of Rights', 'Cautious Statement', 'Comment', 'Condition', 'Declaration', 'Description', 'Estimate', 'Explanans', 'Explanation', 'Explicandum', 'Falsehood', 'Formula', 'Mathematical Statement', 'Misstatement', 'Negation', 'Pleading', 'Prediction', 'Proposition', 'Quotation', 'Recital', 'Remark', 'Representation', 'Reservation', 'Restatement', 'Rhetorical Question', 'Solution', 'Thing', 'Truth', 'Understatement', 'Value Statement', 'Word']

a brief statement that presents the main points in a concise form


```

def disambiguate(sent):
    sentence_for_bert = sent.replace("***", " [TGT] ")
    sentence_for_bert = " ".join(sentence_for_bert.split())
    # try:
    sense, meaning, answer = get_sense(sentence_for_bert)
    if sense is not None:
        distractors = get_distractors_wordnet(sense, answer)
    else:
        distractors = ["Word not found in Wordnet. So unable to extract distractors."]
    sentence_for_T5 = sent.replace("***", " ")
    sentence_for_T5 = " ".join(sentence_for_T5.split())
    ques = get_question(sentence_for_T5, answer)
    return ques, answer, distractors, meaning

sentence = ("If you need more information about your medical condition or your treatment, read the package **leaflet** or contact your doctor or pharmacist.")

question, answer, distractors, meaning = disambiguate(sentence)

print (question)
print (answer)
print (distractors)
print (meaning)

```

OUTPUT :

```

What is the name of the package that contains information about your medical condition?</s>

leaflet

['Appointment Book', 'Authority', 'Bestiary', 'Booklet', 'Catalog', 'Catechism', 'Copybook', 'Curiosa', 'Formulary', 'Phrase Book', 'Playbook', 'Pop-up Book', 'Prayer Book', 'Reference Book', 'Review Copy', 'Songbook', 'Storybook', 'Textbook', 'Tome', 'Trade Book', 'Workbook', 'Yearbook']

a small book usually having a paper cover

```

```

def disambiguate(sent):
    sentence_for_bert = sent.replace("***", " [TGT] ")
    sentence_for_bert = " ".join(sentence_for_bert.split())
    # try:
    sense, meaning, answer = get_sense(sentence_for_bert)
    if sense is not None:
        distractors = get_distractors_wordnet(sense, answer)
    else:
        distractors = ["Word not found in Wordnet. So unable to extract distrac
tors."]
    sentence_for_T5 = sent.replace("***", " ")
    sentence_for_T5 = " ".join(sentence_for_T5.split())
    ques = get_question(sentence_for_T5, answer)
    return ques, answer, distractors, meaning

sentence = ("It explains how the committee for Medicinal Products for Human
Use assessed the studies performed, to reach their **recommendations** on
how to use the medicine.")

question, answer, distractors, meaning = disambiguate(sentence)

print (question)
print (answer)
print (distractors)
print (meaning)

```

OUTPUT:

```

What does the committee of Medicinal Products for Human Use make?</s>
recommendations
['Admonition', 'Indication', 'Recommendation']
something (as a course of action) that is recommended as advisable

```

CHAPTER 4 INFERENCE

4.1. SUMMARY

Word Sense Disambiguation (WSD) is a task concerned with analysing the correct sense of an ambiguous word. To minimise the gap between humans and computers and to provide better interfacing, there is need to improve the accuracy of the systems for this task.

For example, if we consider the word *plant* has different senses which can be analysed only by looking at the context. This is effortless for humans as we are well aware from the day to day learning experience. WSD aims at enabling computers to do the same.

If we consider sentences "This is a factory plant that produces cushions" and "We must plant trees regularly". The sense of the word 'plant' is industrial site and a living organism respectively. We can observe that word plant is not sufficient to determine the correct sense but if we consider the surrounding words, the sense becomes clear. Using WSD, the first sentence can be **sense-tagged** as /factory plant / produce cushions.

In this report, we have addressed the need to increase accuracy of existing WSD systems, and to set forth our presumed model based on supervised approach. The model uses a two level algorithm that uses our improvised system at the first level and Lesk algorithm at second level to maximize accuracy.

4.2. CONCLUSION

Considering the results from the study in previous section we have done the precise analysis about the overall performance of each algorithm. One main inference that can be concluded from the evaluation is that supervised systems clearly surpass knowledge-based models. The main disambiguation trace is provided by its preceding and immediate following word, which occurs before and after the particular sense. For knowledge-based approach the Wordnet first sense criterion proves still to be extremely rigid.

This paper focuses on integrated evaluation framework for all-words WSD. This study relies on evaluation dataset taken from Senseval-2015-en and automatically sense-interpreted corpora. In this evaluation framework dataset has xml format, sense inventory (i.e., WordNet 3.0), which reduces the complexity of the task to assess models. Also it does a fair comparison among proposed algorithms. Supervised systems attains the more encouraging results.

4.3. FUTURE WORK

According to our analysis, we anticipate that further scanning the complete dataset with disambiguating each word of each sentence and then comparing the results with keys present in wordnet. Moreover study among more datasets of SemCor & SemEval that share common format and concatenating all the datasets into a single framework. As far as knowledge-based approach is concerned, enhancing knowledge resources with connotation connections for non-nominal mentions can be an important step further upgrading its performance.

APPENDIX : LIST OF PUBLICATIONS**Publication Details :**

JOURNAL NAME : International Journal of Special Education

ISSN NO. : 0827-3383

MANAGER : Institute for Scientific and Engineering Research (ISER)

STATUS : ACCEPTED

DATE : 5 May 2022

Conference Details :

EVENT NAME : International Conference on Engineering & Technology, Computer, Basic & Applied Sciences (ICETCBAS-22)

EVENT PLACE : Kolkata, India

DATE : 23 April 2022

STATUS : CERTIFIED for presentation

MODE OF PRESENTATION : Online

REFERENCES

- [1]. Huang, Luyao, Chi Sun, Xipeng Qiu, and Xuanjing Huang. "GlossBERT: BERT for word sense disambiguation with gloss knowledge." *arXiv preprint arXiv:1908.07245* (2019).
- [2]. Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, Valencia, Spain. Association for Computational Linguistics.
- [3]. July 17, 2021, Complete Guide To Bidirectional LSTM (With Python Codes) <https://analyticsindiamag.com/complete-guideto-bidirectional-lstm-with-python-codes/>
- [4]. Niloofar Ranjbar and Hossein Zeinali. 2021. Lotus at SemEval-2021 Task 2: Combination of BERT and Paraphrasing for English Word Sense Disambiguation. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, Online. Association for Computational Linguistics.
- [5]. Banerjee, Satanjeev, and Ted Pedersen. "An adapted Lesk algorithm for word sense disambiguation using WordNet." 2002, In *International conference on intelligent text processing and computational linguistics*. Springer, Berlin, Heidelberg.
- [6]. Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, 2018, "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805*.
- [7]. Lesk, Michael, 1986, "Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone." In *Proceedings of the 5th annual international conference on Systems documentation*.