Experiment 8

Aim

To implement Word Sense Disambiguation using Synset

Theory

Word Sense Disambiguation is an important method of NLP by which the meaning of a word is determined, which is used in a particular context. NLP systems often face the challenge of properly identifying words, and determining the specific usage of a word in a particular sentence has many applications.

Word Sense Disambiguation basically solves the ambiguity that arises in determining the meaning of the same word used in different situations.

How to implement WSD?

There are four main ways to implement WSD.

- **Dictionary- and knowledge-based methods**
    - These methods rely on text data like dictionaries, thesaurus, etc. It is based on the fact that words that are related to each other can be found in the definitions. The popularly used Lesk method, which is a seminal dictionary-based method.
- **Supervised methods**
    - In this type, sense-annotated corpora are used to train machine learning models. But a problem that may arise is that such corpora are very tough and time-consuming to create.
- **Semi-supervised Methods**
    - Due to the lack of such corpus, most word sense disambiguation algorithms use semi-supervised methods. The process starts with a small amount of data, which is often manually created.
    - This is used to train an initial classifier. This classifier is used on an untagged part of the corpus, to create a larger training set. Basically, this method involves bootstrapping from the initial data, which is referred to as the seed data.
    - Semi-supervised methods thus, use both labeled and unlabelled data.

- **Unsupervised Methods**
    - Unsupervised Methods pose the greatest challenge to researchers and NLP professionals. A key assumption of these models is that similar meanings and senses occur in a similar context. They are not dependent on manual efforts, hence can overcome the knowledge acquisition deadlock.

Lesk Algorithm

Lesk Algorithm is a classical Word Sense Disambiguation algorithm introduced by Michael E. Lesk in 1986.

The Lesk algorithm is based on the idea that words in a given region of the text will have a similar meaning. In the Simplified Lesk Algorithm, the correct meaning of each word context is found by getting the sense which overlaps the most among the given context and its dictionary meaning.

Code

```
import nltk

nltk.download("omw-1.4")
nltk.download("wordnet")
from nltk.corpus import wordnet as wn
from nltk.wsd import lesk

sentence = ["I", "went", "to", "the", "track", "to", "watch", "the", "car", "race", "."]

print(lesk(sentence, "track", "n"), "\n")
print(lesk(sentence, "track"), "\n")

syns = wn.synsets("track")
print(syns, "\n")

# print the word
print(syns[0].lemmas()[0].name(), "\n")

from nltk.corpus import wordnet as wn

for ss in wn.synsets("track"):
    print(ss, ss.definition())
print("\n")

print([(s, s.pos()) for s in wn.synsets("can")], "\n")

sentence = "I can kill seventy monsters in a minute".split()
print(lesk(sentence, "can"), "\n")
print(lesk(sentence, "can", pos="v"), "\n")

print(lesk("Ram adores Site".split(), "adores", synsets=[]))
```

Output

```
Aryan D:\..\..\..\NLP env:venv >>> python 8.py
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\Aryan\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Aryan\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Synset('track.n.11')

Synset('track.n.11')

[Synset('path.n.04'), Synset('lead.n.03'), Synset('track.n.03'), Synset('racetrack.n.01'), Synset('cut.n.08'), Synset('track.n.06'), Sy
nset('track.n.07'), Synset('track.n.08'), Synset('track.n.09'), Synset('track.n.10'), Synset('track.n.11'), Synset('track.v.01'), Synse
t('track.v.02'), Synset('chase.v.01'), Synset('traverse.v.01'), Synset('track.v.05')]

path

Synset('path.n.04') a line or route along which something travels or moves
Synset('lead.n.03') evidence pointing to a possible solution
Synset('track.n.03') a pair of parallel rails providing a runway for wheels
Synset('racetrack.n.01') a course over which races are run
Synset('cut.n.08') a distinct selection of music from a recording or a compact disc
Synset('track.n.06') an endless metal belt on which tracked vehicles move over the ground
Synset('track.n.07') (computer science) one of the circular magnetic paths on a magnetic disk that serve as a guide for writing and rea
ding data
Synset('track.n.08') a groove on a phonograph recording
Synset('track.n.09') a bar or pair of parallel bars of rolled steel making the railway along which railroad cars or other vehicles can
roll
Synset('track.n.10') any road or path affording passage especially a rough one
Synset('track.n.11') the act of participating in an athletic competition involving running on a track
Synset('track.v.01') carry on the feet and deposit
```

```
Synset('path.n.04') a line or route along which something travels or moves
Synset('lead.n.03') evidence pointing to a possible solution
Synset('track.n.03') a pair of parallel rails providing a runway for wheels
Synset('racetrack.n.01') a course over which races are run
Synset('cut.n.08') a distinct selection of music from a recording or a compact disc
Synset('track.n.06') an endless metal belt on which tracked vehicles move over the ground
Synset('track.n.07') (computer science) one of the circular magnetic paths on a magnetic disk that serve as a guide for writing and rea
ding data
Synset('track.n.08') a groove on a phonograph recording
Synset('track.n.09') a bar or pair of parallel bars of rolled steel making the railway along which railroad cars or other vehicles can
roll
Synset('track.n.10') any road or path affording passage especially a rough one
Synset('track.n.11') the act of participating in an athletic competition involving running on a track
Synset('track.v.01') carry on the feet and deposit
Synset('track.v.02') observe or plot the moving path of something
Synset('chase.v.01') go after with the intent to catch
Synset('traverse.v.01') travel across or pass over
Synset('track.v.05') make tracks upon


[(Synset('can.n.01'), 'n'), (Synset('can.n.02'), 'n'), (Synset('can.n.03'), 'n'), (Synset('buttocks.n.01'), 'n'), (Synset('toilet.n.02'
), 'n'), (Synset('toilet.n.01'), 'n'), (Synset('can.v.01'), 'v'), (Synset('displace.v.03'), 'v')]

Synset('can.v.01')

Synset('can.v.01')

None
```

Conclusion

Thus, we implemented Word Sense Disambiguation in Python using Synset Lesk.