NATURAL LANGUAGE PROCESSING
AI 829

PROJECT REPORT

HANDLING LEXICAL AMBIGUITY IN PROMPTS

SUBMITTED BY

DRISHTI GUPTA
MT2023099

AYUSHI PRASAD
MT2023145

INTRODUCTION:

For many users of natural language processing (NLP), it can be challenging to obtain concise, accurate and precise answers to prompts in LLM models like chatGPT , Gemini etc. These systems enable users to give prompts and receive feedback in the form of quick answers to questions posed in natural language, rather than in the form of lists of documents delivered by search engines. This task is challenging and involves complex semantic annotation and knowledge representation.The problem statement revolves around addressing lexical ambiguity in prompts within natural language processing (NLP) using WSD method.

★ IMPORTANCE OF THE PROBLEM

Addressing ambiguity in question answering is crucial for several reasons:

- Improving ambiguity is crucial as it may lead to misinterpretation of user queries resulting in incorrect answers.
- Natural language is inherently ambiguous, and addressing this challenge is essential for practical applications of NLP, such as customer support bots, and information retrieval systems.
- Ambiguity handling improves the precision of information retrieval systems, allowing users to access the information they need more efficiently.
- Misinterpretation of ambiguous queries can lead to miscommunication between users and the system. Addressing ambiguity helps minimize communication gaps and ensures a more accurate exchange of information.

MANDATE 2 and 3

In this mandate we have done preprocessing of the dataset which involves Part of Speech tagging(POS Tagging),stopword removal,tokenization,lower casing all the characters,lemmatization.

LIBRARIES AND DICTIONARY USED

1) Data Purifier: We used this library to perform tasks like lower casing,removal of white spaces etc.

2) nltk: We used this library to perform tokenization, pos tagging and to import wordnet.

3) Wordnet:WordNet is a lexical database of semantic relations between words that links words into semantic relations including synonyms, hyponyms, and meronyms. The synonyms are grouped into *synsets* with short definitions and usage examples. It can thus be seen as a combination and extension of a dictionary and thesaurus. While it is accessible to human users via a web browser,its primary use is in automatic text analysis and artificial intelligence applications.

4) Synset:Synset is a special kind of a simple interface that is present in NLTK to look up words in WordNet. Synset instances are the groupings of synonymous words that express the same concept. Some of the words have only one Synset and some have several.

STEPS

a) preprocessing

1)STOP WORD REMOVAL: It is better to filter words like out a, an, the, in etc. which do not give any semantic meaning to the sentence. Here, we use stop word list stores in NLTK in python.

2)TOKENIZATION: Tokenization is the process of splitting text into smaller units such as words or sentences. We did tokenization because POS tagging requires words to get tag : noun,adjective,verb and adverb.

3)POS TAGGING: To classify the words on the basis of part of speech category, part of speech tagging is done. Part of speech tagging classifies the content words. Tags include noun, adjective, verb and adverb.

```
[11] import nltk
     try:
         pure_df = pure.df
         print(pure_df)
     except ValueError as e:
         print(f"Error: {e}")
     nltk.download('punkt')
     nltk.download('averaged_perceptron_tagger')

     pure_df["prompts"] = pure_df["prompts"].astype(str)

     # Function to perform tokenization and POS tagging
     def tokenize_and_tag(text):
       sentences = nltk.sent_tokenize(text)
       tagged_sentences = []
       for sentence in sentences:
         words = nltk.word_tokenize(sentence)
         tagged = nltk.pos_tag(words)
         tagged_sentences.append(tagged)
       return tagged_sentences

     # Apply the function to each row in the "prompt" column and store the results in a new column
     pure_df["tagged_sentences"] = pure_df["prompts"].apply(tokenize_and_tag)

     print(df)
```

	prompts	ambiguous_words	tagged_sentences
0	write press release announcing major breakthrough	breakthrough	[[(write, JJ), (press, NN), (release, NN), (an...
1	design website company offering new service	service	[[(design, NN), (website, JJ), (company, NN), ...
2	create horror story character haunted dark	dark	[[(create, NN), (horror, NN), (story, NN), (ch...
3	imagine whimsical creature fur changes color	color	[[(imagine, JJ), (whimsical, JJ), (creature, N...
4	generate soundscape evokes feeling lost	lost	[[(generate, NN), (soundscape, NN), (evokes, V...

b) Steps followed after preprocessing :

1)After doing the POS Tagging of each prompt we created a separate column of ambiguous word tag which contains the tag of ambiguous word.

```
[16]
# Function to perform POS tagging for each row in the DataFrame
def tag_ambiguous_word(row):
    prompt = row['prompts']
    ambiguous_word = row['ambiguous_words']

    # Tokenize the prompt sentence
    tokens = word_tokenize(prompt)

    # Perform POS tagging on the tokens
    tagged_tokens = pos_tag(tokens)

    # Find the POS tag for the ambiguous word
    ambiguous_word_tag = None
    for token, tag in tagged_tokens:
        if token == ambiguous_word:
            ambiguous_word_tag = tag
            break

    return ambiguous_word_tag

# Apply the function to each row of the DataFrame to tag the ambiguous words
df['ambiguous_word_tag'] = df.apply(tag_ambiguous_word, axis=1)

print(df)
```

	prompts	ambiguous_words	tagged_sentences	ambiguous_word_tag
0	write press release announcing major breakthrough	breakthrough	[[('write', 'JJ'), ('press', 'NN'), ('release'...	NN
1	design website company offering new service	service	[[('design', 'NN'), ('website', 'JJ'), ('compa...	NN
2	create horror story character haunted dark	dark	[[('create', 'NN'), ('horror', 'NN'), ('story'...	NN
3	imagine whimsical creature fur changes color	color	[[('imagine', 'JJ'), ('whimsical', 'JJ'), ('cr...	NN
4	generate soundscape evokes feeling lost	lost	[[('generate', 'NN'), ('soundscape', 'NN'), ('...	VBN

2)Next we found out all possible senses of the tagged ambiguous word From wordnet with the help of wordnet.synsets and created a column word_senses which contains all senses of ambiguous word separated by comma. Then we found out the definitions of each sense from wordnet.

```python
# Function to map NLTK POS tags to WordNet POS tags
def nltk_to_wordnet_pos(nltk_tag):
    if nltk_tag.startswith('J'):
        return wordnet.ADJ
    elif nltk_tag.startswith('V'):
        return wordnet.VERB
    elif nltk_tag.startswith('N'):
        return wordnet.NOUN
    elif nltk_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None

# Function to find synsets (senses) of ambiguous words by their POS tag from WordNet
def find_word_senses(word, tag):
    wn_tag = nltk_to_wordnet_pos(tag)
    if wn_tag:
        synsets = wordnet.synsets(word, pos=wn_tag)
        return synsets
    else:
        return []

# Apply the function to the DataFrame to find the synsets for each ambiguous word
df['word_senses'] = df.apply(lambda row: find_word_senses(row['ambiguous_words'], row['ambiguous_word_tag']), axis=1)
```

	prompts	ambiguous_words	tagged_sentences	ambiguous_word_tag	word_senses
0	write press release announcing major breakthrough	breakthrough	[[('write', 'JJ'), ('press', 'NN'), ('release'...	NN	[Synset('discovery.n.03'), Synset('breakthroug...
1	design website company offering new service	service	[[('design', 'NN'), ('website', 'JJ'), ('compa...	NN	[Synset('service.n.01'), Synset('service.n.02'...
2	create horror story character haunted dark	dark	[[('create', 'NN'), ('horror', 'NN'), ('story'...	NN	[Synset('dark.n.01'), Synset('iniquity.n.01'),...
3	imagine whimsical creature fur changes color	color	[[('imagine', 'JJ'), ('whimsical', 'JJ'), ('cr...	NN	[Synset('color.n.01'), Synset('color.n.02'), S...
4	generate soundscape evokes feeling lost	lost	[[('generate', 'NN'), ('soundscape', 'NN'), ('...	VBN	[Synset('lose.v.01'), Synset('lose.v.02'), Syn...

```
def nltk_to_wordnet_pos(nltk_tag):
    if nltk_tag.startswith('J'):
        return wordnet.ADJ
    elif nltk_tag.startswith('V'):
        return wordnet.VERB
    elif nltk_tag.startswith('N'):
        return wordnet.NOUN
    elif nltk_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None


def find_word_senses(word, tag):
    wn_tag = nltk_to_wordnet_pos(tag)
    if wn_tag:
        synsets = wordnet.synsets(word, pos=wn_tag)
        if synsets:
            definitions = [synset.definition() for synset in synsets]
            return definitions
        else:
            return None
    else:
        return None
```

tagged_sentences	ambiguous_word_tag	word_senses
('press', 'NN'), ('release'...	NN	[a productive insight, making an important dis...
('website', 'JJ'), ('compa...	NN	[work done by one person or group that benefit...
), ('horror', 'NN'), ('story'...	NN	[absence of light or illumination, absence of ...
I'), ('whimsical', 'JJ'), ('cr...	NN	[a visual attribute of things that results fro...
, ('soundscape', 'NN'), ('...	VBN	[fail to keep or to maintain; cease to have, e...

3)Then we split the senses(which were separated by comma) into separate rows and preprocessed the senses(tokenization , stopword removal).

```
[29] import pandas as pd

     def split_definitions(row):
         word_senses = row['word_senses']

         # Check if word_senses is a string
         if isinstance(word_senses, str):
             definitions = word_senses.split(',')
         else:
             # If it's not a string (e.g., it's a float), return an empty list
             definitions = []

         return pd.Series(definitions)

     # Apply the function to each row and stack the resulting rows
     expanded_df = df.apply(split_definitions, axis=1).stack().reset_index(level=1, drop=True).reset_index()
     expanded_df.columns = ['original_index', 'word_sense']

     # Merge with the original DataFrame to bring in other columns
     expanded_df = expanded_df.merge(df.drop('word_senses', axis=1), how='left', left_on='original_index', right_index=True)
```

index	original_index	word_sense	prompts	ambiguous_words
0	0	['a productive insight'	write press release announcing major breakthrough	breakthrough
1	0	'making an important discovery'	write press release announcing major breakthrough	breakthrough
2	0	"a penetration of a barrier such as an enemy's defense"]	write press release announcing major breakthrough	breakthrough
3	1	['work done by one person or group that benefits another'	design website company offering new service	service
4	1	'an act of help or assistance'	design website company offering new service	service
5	1	'the act of public worship following prescribed rules'	design website company offering new service	service
6	1	'a company or agency that performs a public service; subject to government regulation'	design website company offering new service	service
7	1	'employment in or work for another'	design website company offering new service	service
8	1	'a force that is a branch of the armed forces'	design website company	service

4)Next we took out the content words(noun,verb,adverb and adjective) from each sense of ambiguous word.

```
[31] from nltk.corpus import stopwords
     from nltk.tokenize import word_tokenize
```

```
[32] import nltk
     nltk.download('stopwords')
     from nltk.corpus import stopwords
     # Function to filter content words from text
     def filter_content_words(text):
         stop_words = set(stopwords.words('english'))
         tokens = word_tokenize(text.lower())  # Tokenize and convert to lowercase
         filtered_tokens = [token for token in tokens if token.isalnum() and token not in stop_words]
         pos_tags = nltk.pos_tag(filtered_tokens)  # Get POS tags for filtered tokens
         content_words = [token for token, tag in pos_tags if tag.startswith(('N', 'V', 'J', 'R'))]
         return ' '.join(content_words)

     # Apply the function to the 'word_sense' column
     expanded_df['content_words'] = expanded_df['word_sense'].apply(filter_content_words)
```

prompts	ambiguous_words	tagged_sentences	ambiguous_word_tag	content_words
ing major ıkthrough	breakthrough	[[('write', 'JJ'), ('press', 'NN'), ('release'...	NN	productive insight
ing major ıkthrough	breakthrough	[[('write', 'JJ'), ('press', 'NN'), ('release'...	NN	important discovery
ing major ıkthrough	breakthrough	[[('write', 'JJ'), ('press', 'NN'), ('release'...	NN	penetration barrier enemy defense
ering new service	service	[[('design', 'NN'), ('website', 'JJ'), ('compa...	NN	done person group benefits
ering new service	service	[[('design', 'NN'), ('website', 'JJ'), ('compa...	NN	act help assistance

MANDATE 4

In this mandate,we made pairs of content words from sense and prompt and calculated the collocation score of each pair and pick the sense which gave the maximum collocation score.We used wikipedia corpus to find the frequency of individual words and words in pairs occurring together with span width of 2187.

LIBRARIES AND DICTIONARY USED:

1) Datasets:Datasets is a library for easily accessing and sharing datasets for Audio, Computer Vision, and Natural Language Processing (NLP) tasks.

2) Counter:Counter class is a special type of object data-set provided with the collections module in Python3. Collections module provides the user with specialized container datatypes, thus, providing an alternative to Python's general-purpose built-ins like dictionaries, lists, and tuples.We have used update function of counter class.
The Counter.update() method updates the counts of elements in a Counter object based on the elements provided in an iterable or another Counter object. It takes an iterable or a mapping (such as another Counter) as its argument and increments the counts of elements accordingly.

STEPS:

1)We got the content words of senses and content words of prompts and made pairs of each word from content words of senses and content words of prompts.

·riginal_index	word_sense	prompts	ambiguous_words	tagged_sentences	ambiguous_word_tag	content_words_sense	content_words_prompts
0	['a productive insight'	Write a press release announcing a major break...	breakthrough	[[('Write', 'VB'), ('a', 'DT'), ('press', 'NN'...	NN	productive insight	write press release announcing major breakthrough
0	'making an important discovery'	Write a press release announcing a major break...	breakthrough	[[('Write', 'VB'), ('a', 'DT'), ('press', 'NN'...	NN	important discovery	write press release announcing major breakthrough
0	"a penetration of a barrier such as an enemy'...	Write a press release announcing a major break...	breakthrough	[[('Write', 'VB'), ('a', 'DT'), ('press', 'NN'...	NN	penetration barrier enemy defense	write press release announcing major breakthrough

·x	word_sense	prompts	ambiguous_words	tagged_sentences	ambiguous_word_tag	content_words_sense	content_words_prompts	word_pairs
0	['a productive insight'	Write a press release announcing a major break...	breakthrough	[[('Write', 'VB'), ('a', 'DT'), ('press', 'NN'...	NN	productive insight	write press release announcing major	[(productive, write), (productive, press), (pr...
0	'making an important discovery'	Write a press release announcing a major break...	breakthrough	[[('Write', 'VB'), ('a', 'DT'), ('press', 'NN'...	NN	important discovery	write press release announcing major	[(important, write), (important, press), (impo...
0	"a penetration of a barrier such as an enemy'	Write a press release announcing a major	breakthrough	[[('Write', 'VB'), ('a', 'DT'), ('press', 'NN'...	NN	penetration barrier enemy defense	write press release announcing major	[(penetration, write), (penetration, press), (...

2)Downloaded the wikipedia corpus dataset.Wikipedia dataset contains title column and text column which contains text corresponding to title.

```
split_name = 'train'
dataset_split = dataset[split_name]

# Extract a subset of examples (e.g., first 1000 examples)
subset_size = 1000
subset_dataset = dataset_split[:subset_size]

# Convert subset_dataset to a Pandas DataFrame
df_subset = pd.DataFrame(subset_dataset)
```

Challenge faced: We took subset of wikipedia dataset after downloading the entire corpus dataset as it was taking a lot of time to find the frequency of words in the entire corpus and it failed at the end.So we created a subset of 1000 examples from wikipedia corpus.

3)Then we defined word_occurrences function to calculate the frequency of individual words present in the pairs in the word_pair column.

ıbiguous_words	tagged_sentences	ambiguous_word_tag	content_words_sense	content_words_prompts	word_pair	word_pair_list	Frequency
breakthrough	[[('Write', 'VB'), ('a', 'DT'), ('press', 'NN'...	NN	productive insight	write press release announcing major	(productive, write)	[productive, write]	{'productive': 45, 'write': 251}
breakthrough	[[('Write', 'VB'), ('a', 'DT'), ('press', 'NN'...	NN	productive insight	write press release announcing major	(productive, press)	[productive, press]	{'productive': 45, 'press': 104}
breakthrough	[[('Write', 'VB'), ('a', 'DT'), ('press', 'NN'...	NN	productive insight	write press release announcing major	(productive, release)	[productive, release]	{'productive': 45, 'release': 273}

Challenge faced: Initially we had 100 prompts in our dataset but as word_occurrences function was taking a lot of time to and it failed at the end so we decided to reduce our dataset to 50 prompts.Now we are working on 50 prompts only.

4)Then we defined count_word_pairs function to count the occurrences of words occurring together with span width of 2187 present in the pairs in the word_pair column.

·d_sentences	ambiguous_word_tag	content_words_sense	content_words_prompts	word_pair	word_pair_list	Frequency	word_pair	freq
rite', 'VB'), ('a', 'DT'), ('press', 'NN'...	NN	productive insight	write press release announcing major	(productive, write)	[productive, write]	{'productive': 45, 'write': 251}	productive write	3.0
rite', 'VB'), ('a', 'DT'), ('press', 'NN'...	NN	productive insight	write press release announcing major	(productive, press)	[productive, press]	{'productive': 45, 'press': 104}	productive press	2.0
rite', 'VB'), ('a', 'DT'), ('press', 'NN'...	NN	productive insight	write press release announcing major	(productive, release)	[productive, release]	{'productive': 45, 'release': 273}	productive release	4.0

Challenge faced: Initially when we took span width of 3 we got all counts to be zero.But as we increased our span width to 9,27,81… we got most of the count values to be zero.At span width of 2187 we got very less non zero values.so we picked 2187.Span width 3 and later values didn’t work because we took a subset of 100 examples of wikipedia corpus.

5)Now we found the word count in the subset of wikipedia corpus we took which came out to be 3544633 words.

```
# Assuming you have a DataFrame named wiki_data

# Option 1: Using apply and split
word_count = wiki_data['text'].apply(lambda x: len(x.split())).sum()

print("Total number of words in the 'text' column:", word_count)
```

```
Total number of words in the 'text' column: 3544633
```

6)Now we applied the collocation score formula and created the function calculate_formula to calculate the score.

$$collocation\_score(w, w\prime) = \frac{log(\frac{(x*sizeCorpus)}{(w*w\prime*span)})}{log(2)}$$

```
import numpy as np
def calculate_formula(x,w1,w2):
    try:
        result = np.log((x * 3544633) / (w1 * w2 * 2187)) / np.log(2)
    except ZeroDivisionError:
        result = 0
    return result
```

rd_pair_list	Frequency	word_pair.1	freq	Frequency_new	Frequency_values	Frequency_value_1	Frequency_value_2	collocation_score
['productive', 'write']	{'productive': 45, 'write': 251}	productive write	3.0	{'productive': 45, 'write': 251}	[45, 251]	45.0	251.0	-1.215967
['productive', 'press']	{'productive': 45, 'press': 104}	productive press	2.0	{'productive': 45, 'press': 104}	[45, 104]	45.0	104.0	-0.529825
['productive', 'release']	{'productive': 45, 'release': 273}	productive release	4.0	{'productive': 45, 'release': 273}	[45, 273]	45.0	273.0	-0.922143

Conclusion:

As a result of encountering challenges during the aforementioned steps, we were constrained to operate with smaller datasets, which unfortunately yielded suboptimal outcomes.However, if we have access to more powerful computers, we can analyze the entire Wikipedia collection and use a larger dataset of prompts.This advancement is expected to significantly enhance our results, offering a more comprehensive and impactful analysis.