1. Contradiction: A situation when there is disagreement on the same aspect. We can also define contradiction as given the same aspect, if there are two statements which can not be true at the same time, we say these statements contradict each other.
2. Disagreement: When two statements do not completely negate each other on the same aspect. Disagreement is a broader term, a contradiction is always a disagreement, but a disagreement can or can not be a contradiction.
3. Disagreement with Contradiction: When two statements for the same aspects can not be true simultaneously.
4. Disagreement without Contradiction: When two statements address different aspects, they might disagree but there is no contradiction.

   Example of **disagreement with contradiction**: Let the aspect be: Technical correctness of a research paper.
   Now suppose there are 2 reviewers, reviewer one says “The technical correctness of the paper is very bad” and reviewer two says “The technical correctness of the paper is good”. The two reviewers have made such statements which can not be true at the same time, hence we can say that the statements contradict each other and it can be called a disagreement with contradiction.

   Example of **disagreement without contradiction:** Let the aspect be: Clarity of research methodology.
   Suppose one reviewer said that “The research methodology is very clear in explaining the data collection process” and the second reviewer said that “The research methodology lacks clarity in detailing the statistical analysis techniques.” So here even though the overall aspect is clarity of research methodology, the reviews made are evaluating it on different grounds/points. This is called disagreement without contradiction.

Chalk out a plan (broad algorithm) to detect (and extract) contradiction in a paragraph using mostly classical NLP based:

We will define some keywords for aspects ourselves, then we will try to find sentences in reviews which will relate to these aspects and make a pair of those. Then we will apply sentiment analysis to check if the sentiment is positive or negative for that aspect, if they are opposite, it means that they contradict each other, it’ll count as disagreement with contradiction. Now if one sentence is positive and other is neutral or one is negative and other is neutral we can say it as disagreement without contradiction.

Pseudocode:

```
# Step 1: Define aspects and associated keywords ASPECTS = { "methodology":
["methodology", "clear", "confusing"],
"novelty": ["novel", "original", "new"],
 "experiments": ["experiment", "design", "results"],
}


# Step 2: Preprocess the paragraph
sentences = tokenize(paragraph)
normalized_sentences = normalize(sentences)
sentence_pairs = []

# Step 3: Extract sentences by aspect
FOR aspect, keywords IN ASPECTS:
sentences_for_aspect = []
FOR sentence IN normalized_sentences:
        IF contains_keywords(sentence, keywords):
                sentences_for_aspect.append(sentence)
# Create pairs of sentences for this aspect
FOR i FROM 0 TO LENGTH(sentences_for_aspect) - 1:
        FOR j FROM i + 1 TO LENGTH(sentences_for_aspect):
                sentence_pairs.append((sentences_for_aspect[i], sentences_for_aspect[j]))

results = []
# Step 4: Perform Sentiment Analysis on each pair
FOR pair IN sentence_pairs:
        sent1 = pair[0]
        sent2 = pair[1]
        sentiment1 = analyze_sentiment(sent1)
        sentiment2 = analyze_sentiment(sent2)

# Step 5: Classify disagreements
IF sentiments_are_opposite(sentiment1, sentiment2):
        results.append((sent1, sent2, "disagreement with contradiction"))
ELSE IF is_disagreement_without_contradiction(sentiment1, sentiment2):
        results.append((sent1, sent2, "disagreement without contradiction"))

# Step 6: Output results RETURN results
```

For the CLM task, annotated data is crucial to effectively train our model. However, we currently do not have access to pre-annotated datasets for this specific task. To overcome this challenge, we propose using GPT to automatically generate the necessary annotations. This will allow us to annotate our data at scale, ensuring that it is suitable for training purposes. Once we have this annotated dataset, we can fine-tune a model, such as BERT, to perform the task of contradiction identification and analysis.
The fine-tuned model will be responsible for handling the following key tasks in CLM:

Detecting Contradictions Between Reviews: The model will determine whether two given reviews are contradictory in nature. This is important for identifying conflicting opinions or statements, especially in user-generated content like product reviews or social media posts.

Identifying Points of Contradiction (Aspects): In addition to detecting contradictions, the model will also localize the specific aspects or points where the contradictions occur. For instance, one review might praise a product's durability while another criticizes it for being flimsy. The model will highlight these conflicting aspects.

Classifying the Type of Contradiction: The model will categorize contradictions into three main types:

- **Contradiction with Disagreement**: This occurs when two reviews express clearly opposing viewpoints, such as one praising a product's quality while the other criticizes it.
- **Contradiction without Disagreement**: In this case, the contradiction exists but is due to factors like misunderstanding, differing contexts, or personal preferences, rather than outright disagreement.
- **Contradiction with Agreement**: Although contradictory in terms of details or specifics, the reviews may still agree on the overall sentiment or conclusion, such as both recommending a product despite highlighting different issues or benefits.

Pseudocode:

```
# Step 1: Define aspects and associated keywords

ASPECTS = {

    "durability": ["durable", "long-lasting", "fragile"],

    "price": ["price", "expensive", "cheap", "affordable"],

    "quality": ["quality", "good", "bad", "high-quality", "poor"],
```

```
    "design": ["design", "look", "appearance", "style"],
}

# Step 2: Preprocess the reviews
review1_sentences = tokenize(Review1)
review2_sentences = tokenize(Review2)
normalized_review1 = normalize(review1_sentences)
normalized_review2 = normalize(review2_sentences)
sentence_pairs = []

# Step 3: Extract sentences by aspect
FOR aspect, keywords IN ASPECTS:
    sentences_for_aspect_review1 = []
    sentences_for_aspect_review2 = []

    FOR sentence IN normalized_review1:
        IF contains_keywords(sentence, keywords):
            sentences_for_aspect_review1.append(sentence)

    FOR sentence IN normalized_review2:
        IF contains_keywords(sentence, keywords):
            sentences_for_aspect_review2.append(sentence)
```

```
    # Create pairs of sentences for this aspect
    FOR i FROM 0 TO LENGTH(sentences_for_aspect_review1):
        FOR j FROM 0 TO LENGTH(sentences_for_aspect_review2):
            sentence_pairs.append((sentences_for_aspect_review1[i],
sentences_for_aspect_review2[j]))

results = []

# Step 4: Perform Sentiment Analysis on each pair
FOR pair IN sentence_pairs:
    sent1 = pair[0]
    sent2 = pair[1]
    sentiment1 = analyze_sentiment(sent1)
    sentiment2 = analyze_sentiment(sent2)

    # Step 5: Classify contradictions and types
    IF sentiments_are_opposite(sentiment1, sentiment2):
        results.append((sent1, sent2, "contradiction with disagreement"))
    ELSE IF sentiments_agree_but_details_vary(sentiment1, sentiment2):
        results.append((sent1, sent2, "contradiction with agreement"))
    ELSE IF different_context_but_no_disagreement(sentiment1, sentiment2):
        results.append((sent1, sent2, "contradiction without disagreement"))
# Step 6: Output results
RETURN results
```

References :

[1] S. Kumar, T. Ghosal, and A. Ekbal, "When Reviewers Lock Horn: Finding Disagreement in Scientific Peer Reviews," *arXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2310.18685.