Monday, June 2, 2025 20:07

```python
def process_pharma_matches(self, combined_matches):
    """
    Process the combined matches from keyword and LLM detection
    to extract pharmaceutical terms and get detailed information.

    Args:
        combined_matches: List of text strings that potentially contain pharmaceutical references

    Returns:
        Dictionary with detailed pharmaceutical information for each match
    """
    import re

    get_logger().info(f"Processing {len(combined_matches)} matches for pharma information")

    # Results dictionary to store pharma info for each match
    results = {}

    # Common words to exclude from potential drug name candidates
    common_words = {"with", "that", "this", "from", "have", "your", "what", "when", "where",
                    "which", "about", "there", "their", "they", "them", "these", "those"}

    for match in combined_matches:
        get_logger().info(f"Processing match: {match[:50]}...")

        # 1. Extract potential pharma terms
        potential_terms = set()

        # Word-based extraction (length > 3, not common words)
        words = re.findall(r'\b[a-zA-Z]{4,}\b', match.lower())
        for word in words:
            if word not in common_words:
                potential_terms.add(word)

        # N-gram extraction for multi-word drug names
        word_list = match.lower().split()

        # Add 2-grams and 3-grams that might be drug names
        for i in range(len(word_list) - 1):
            bigram = " ".join(word_list[i:i+2])
            if not any(w in common_words for w in bigram.split()):
                potential_terms.add(bigram)

        for i in range(len(word_list) - 2):
            trigram = " ".join(word_list[i:i+3])
            if not any(w in common_words for w in trigram.split()):
                potential_terms.add(trigram)

        # Pattern-based extraction (e.g., "10mg Xanax", "Adderall XR")
        dosage_patterns = re.findall(r'(\b[a-zA-Z]{3,})\s+\d+\s*(?:mg|mcg|g)\b', match)
        dosage_patterns += re.findall(r'\b\d+\s*(?:mg|mcg|g)\s+(\b[a-zA-Z]{3,})\b', match)
        potential_terms.update(dosage_patterns)

        brand_patterns = re.findall(r'\b([A-Z][a-z]{2,})\s+(?:XR|SR|IR|ER)\b', match)
        potential_terms.update(brand_patterns)

        # 2. Query pharma info API for each term
        match_info = {}
        for term in potential_terms:
            if len(term) > 3:  # Minimum term length
                info = self.get_pharma_info(term)
                if info and info.get("data"):  # Only include if valid data returned
                    match_info[term] = info

        # Store results for this match
        results[match] = {
            "terms": list(potential_terms),
            "pharma_info": match_info
        }
```

```
    return results


# After the line that creates combined_matches in detect_aup_violation:
combined_matches = list(set(keyword_matches + model_matches))
get_logger().info(f"Total {len(combined_matches)} combined matches after deduplication")
# For pharma matches, get detailed info
pharma_details = {}
if "pharma" in aup_violation_categories and combined_matches:
    try:
        pharma_details = self.raptor_client.process_pharma_matches(combined_matches)
        # Store for later use in analyze_texts
        self._pharma_details = pharma_details
    except Exception as e:
        get_logger().error(f"Error getting pharma details: {str(e)}")
return combined_matches
```

约个明天的会
Priority 集成到predefined questions

最简单的text category输入代码，能否找出来violation；集成的事情后面再说；还有多约meeting