Natural Language Processing A1 Report

Kashyap J

1b)

How it works

The scraper collects over 100 articles about the ICC Champions Trophy from Sportstar and Sportskeeda using Selenium and BeautifulSoup.

I implemented Selenium because both websites load content dynamically but use different navigation methods:

- Sportstar uses a "Show More" button that requires JavaScript interaction
- Sportskeeda uses traditional pagination (limited to 2 pages because number of articles crossed 100 here)

Content Collection Process

The scraper follows this approach:

1. Navigates to each source website's Champions Trophy section
2. Identifies and collects article titles and URLs
3. Incorporates random delays between requests to avoid triggering anti-scraping measures
4. Tracks previously collected links to prevent duplicates

I implemented site specific extraction functions rather than a generic function to handle the different HTML structures of each website. This improved reliability and extraction accuracy.

Content Extraction & Cleaning

For each article URL, the scraper:

- Navigates to the page and extracts the main content
- Removes irrelevant elements (advertisements, comment sections, related article blocks)
- Filters out sentences containing phrases like "also read" or "related"
- Removes non-UTF characters and normalizes whitespace
- Validates content quality by checking for minimum length (3 sentences) and completeness. It checks if an article is complete by looking for ":", "Find below", "See the full list", "Here are".

Storing the data

The pre-processed data is organized in a pandas DataFrame and CSV with three columns:

- Title
- URL
- Content

The final dataset is saved to **CTScraped.csv**.

2d)

For the entity resolution task, I implemented a character-level TF-IDF vectorization algorithm combined with cosine similarity measurements. I also considered and tried entity resolution using Jaccard similarity and Levenshtein distance but it did not do as good a job.

I chose the TF-IDF with cosine similarity method because:

- the character level n-grams capture similarity despite minor spelling differences/typos, abbreviations or word order changes common in named entities (e.g., "ICC Champions Trophy" vs "Champions Trophy").
- unlike exact string matching, this method can link entities that look slightly different but refer to the same thing (e.g., 'ICC Champions Trophy' and 'Champions Trophy'). It's also better than Levenshtein distance, which might incorrectly link entities just because they have similar spellings, even if they're unrelated.
- vectorization works better than pairwise string comparisons for large entity sets.

Threshold Selection

I decided the similarity threshold of 0.75 because higher thresholds seemed to give more restrictive results and those below 0.7 had more false positives. Although 0.75 does not do an exceptional job, it did a better job than the values with which I tried it.

This value provided the best balance between being strict enough to avoid false matches while allowing for reasonable variations in entity mentions across different articles.

Implementation Challenges

These articles made it a tough task because of:

1. players are often referred to by different name forms (e.g., "Virat Kohli" vs "Kohli" vs "V. Kohli")
2. organizations appear both as full names and acronyms (e.g., "Board of Control for Cricket in India" vs "BCCI")
3. statistics, dates and other numerical values (e.g,'February 19' is matched with 'February 26' and other such dates based on the word February because of cosine similarity.)

The resolved entities were saved in a new column "EntityResolution" in the DataFrame and exported to "entityResolved.csv"

3a)

I manually annotated a subset of 30 articles using label studio for NER Evaluation. The evaluation revealed the following metrics:

Metric	**Value**
Precision	0.4796
Recall	0.5233
F1-Score	0.4921
Accuracy	0.5233

Table 1: Performance metrics- NER model

- **Precision (0.4796):** Measures how many of the entities predicted by the model were correct. A precision of 47.96% means nearly half of the identified entities were valid, while the rest were false positives.
- **Recall (0.5233):** Indicates how many actual entities the model successfully identified. A recall of 52.33% shows that the model missed almost half of the true entities. This is reflected in the many mismatched entities in the **NER_metrics.txt** file.

- **F1-Score (0.4921):** Balances precision and recall. The F1-score of 49.21% indicates the model struggled to maintain both high precision and high recall.
- **Accuracy (0.5233):** Measures the proportion of entity tokens that were correctly identified. At 52.33%, this indicates the model correctly identified just over half of all actual entities in the articles, implying moderate performance.

3b)

I annotated 15 articles using label studio to evaluate the Coreference Resolution model. These were the results :

Metric	Value
Precision	0.0410
Recall	0.0476
F1-score	0.0441
Accuracy	0.0225

Table 2: Performance metrics- Coref model

- **Precision (0.0410):** The proportion of correctly identified coreference links among all predicted links. At 4.10%, most predicted links were incorrect.
- **Recall (0.0476):** The proportion of actual coreference links that were captured. At 4.76%, the model missed most true coreference relationships.
- **F1-Score (0.0441):** The mean of precision and recall. This low score (4.41%) confirms poor overall performance.
- **Accuracy (0.0225):** The proportion of correctly resolved mentions. At 2.25%, the model rarely assigned mentions to their correct coreference clusters.

These extremely low metrics indicate the model performed poorly at coreference resolution. The model failed to identify most actual coreference relationships and made many incorrect predictions. I think this was because the model was not capable of understanding and linking cricketing terms/relations.

3c)

I analysed the outputs from the NER model using three visualizations: a bar plot of entity distributions, a co-occurrence graph and a word cloud. Each visualization provided unique insights into the data.

Entity Type Distribution

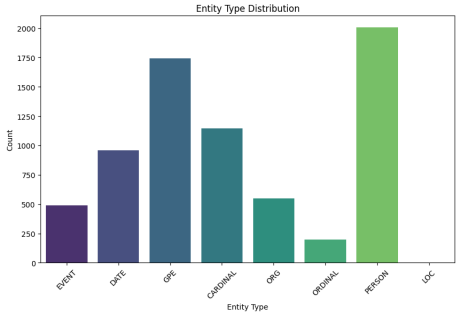

Figure 1: BarGraph Plot

The bar plot highlights the frequency of various entity types identified by the NER model. **PERSON** entities dominate the distribution, followed by **GPE** (Teams, Countries and cities) and **CARDINAL** types. This suggests the dataset is heavily focused on individuals and locations, typical in sports articles discussing teams and players. Entities like **EVENT** and **ORG** appear less frequently but are still significant.

Entity Co-occurrence Graph

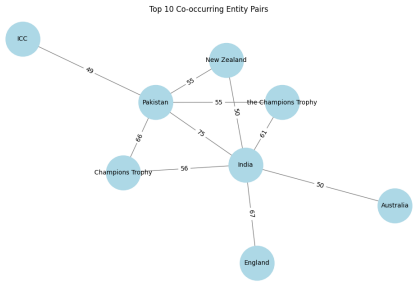

Figure 2: Co-occurrence Graph

The co-occurrence graph showcases relationships between entities based on their co-mentions within the text. Prominent connections include:

- **India** and **Pakistan**, indicating frequent articles involving both teams.
- Strong links between **India** and tournaments like the **Champions Trophy**. This is because the sources were Indian.

- **Pakistan**'s connections with multiple entities, suggesting its centrality in discussions. This is because they are the host nation and the center of attention months before the tournament.

This visualization helps in understanding contextual relationships that go beyond isolated entity mentions.

Word Cloud of Named Entities

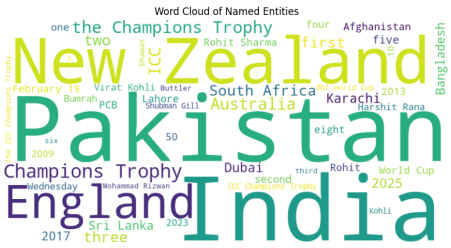

Figure 3: WordCloud

The word cloud emphasizes the most frequently mentioned named entities. **India**, **Pakistan**, and **New Zealand** stand out, indicating their prominence in the dataset. This is because they are the main teams and often appear in articles prior to the tournament.

Additional Insights Beyond NER

- **Entity Sentiment Analysis**: Understanding the sentiment associated with specific entities like players and teams. What opinions of authors say about who the favourites are and who are the player spoken about in good/bad light.
- Identifying specific events (e.g., match outcomes, player milestones) could offer an engaging narrative. This would help with getting a better understanding of which team stands where/ who is winning during a game, etc.

This can be found in **NERAnalyser.ipynb**