Harnessing Open-Source LLMs for Tender Named Entity Recognition

Asim Abbas[1*], Venelin Kovatchev[1], Mark Lee[1], Niloofer Shanavas[2], Mubashir Ali[1]
[1]School of Computer Science, University of Birmingham, Edgbaston, Birmingham, UK
[2]School of Computer Science, University of Birmingham, Dubai Campus, UAE
Correspondence: axa2233@student.bham.ac.uk, asimabbasturi@gmail.com

References

1 Appendices

1.1 Singapore Data Transformation with Augmentation Approach

The data shared with us by Siemens is limited to 30 documents. Beyond this, tender data is not easily accessible, as most organizations and companies treat it as private and confidential information. Additionally, each organization or company follows its own unique pattern and structure for tender management, leading to significant variations in content. To expand our experiments for generalization purpose, we conducted a Google search and identified a publicly available dataset published by Singapore government tenders since 2015, hosted on Kaggle. However, this dataset was in a structured format. To adapt it for our purposes, we designed an algorithm to transform this structured data into an unstructured format suitable for our experiments. Below, we provided a stepwise explanation of the algorithm:

Input: The algorithm takes a structured document as input, which contains nine fields: tender_no, tender_description, agency, award_date, tender_detail_status, supplier_name, awarded_amt, and main_category.

Step 1: We read the given documents row by row and store the data from each field into separate variables (see Algorithm Steps 1 to 9).

Step 2: We defined several patterns, such as Introduction, Award Outcome, and Status Sentence, to guide the transformation process (see Algorithm Steps 10 to 13).

Introduction Patteren: ["$\{agency\}$, a prominent $\{main\_category\}$ organization, has launched Tender No.$\{tender\_no\}$ for $\{description\}$.",
"Announcing Tender No.$\{tender\_no\}$, $\{agency\}(\{main\_category\})$ invites proposals for $\{description\}$.",
"In a call for competitive bids, $\{agency\}(\{main\_category\})$ released Tender No.$\{tender\_no\}$ regarding $\{description\}$.",
"$\{agency\}(\{main\_category\})$ has issued Tender No.$\{tender\_no\}$ seeking submissions for $\{description\}$.",
"With an aim to secure quality services, $\{agency\}$ from $\{main\_category\}$ published Tender No. $\{tender\_no\}$ for $\{description\}$."
]

Award Outcome Patteren-1: ["After a meticulous evaluation, no supplier was awarded as of $\{award\_date\}$.",
"As of $\{award\_date\}$, the tender did not attract any qualifying bids, leaving it unawarded.",
"Despite an extensive review, no supplier met the criteria by $\{award\_date\}$.",
"Ultimately, no supplier was selected, and the tender remained unawarded as of $\{award\_date\}$."
]

Award Outcome Patteren-2: ["The contract was awarded on $\{award\_date\}$ to $\{supplier\}$ with a value of $\{amount\}$.",
"On $\{award\_date\}, \{supplier\}$ secured the contract, which is valued at $\{amount\}$.",
"Following a thorough evaluation, the award was granted on $\{award\_date\}$ to $\{supplier\}$ for $\{amount\}$.",
"$\{supplier\}$ was chosen on $\{award\_date\}$ to fulfill the tender with a total value of $\{amount\}$."
]

Status Pattern: ["Tender status is recorded as $\{status\}$.",
"The current status of the tender is: $\{status\}$.",
"Status update: $\{status\}$",
"This tender holds the status: $\{status\}$"]

Step 3: We apply a conditional check:
If the value of the $status$ variable is either "awarded to no suppliers" or "no supplier

awarded", we randomly select a value from the *Award Outcome Pattern-1* list and store it in a variable called $award\_text$ (see Algorithm Steps 14-15).

Similarly, if the $supplier$ variable contains the value "unknown", we store the following pattern in the $award\_text$ variable: ["Although an award decision was made on $\{award\_date\}$, the supplier's details remain undisclosed."] (see Algorithm Steps 16-17).

If neither of the above conditions is true, we randomly select a value from the *Award Outcome Pattern-2* list and store it in the $award\_text$ variable.

Step 4: In this step, we randomly select one sentence each from the *Introduction Pattern* and *Status Pattern* lists and store them in separate variables (see Algorithm Steps 21-23).

Step 5: Finally, we shuffle all the selected variables containing the chosen sentences. We then concatenate these three variables and store the result in a variable called $final\_text$, which is then returned as the output.

The augmented dataset is available on the GitHub along with implemented algorithm.

Algorithm 1 Convert Structured Tender Data into Unstructured Text Apply Data Augmentation Technique

Require: row with fields: `tender_no`, `tender_description`, `agency`, `award_date`, `tender_detail_status`, `supplier_name`, `awarded_amt`, `main_category`

Extract fields from row:
    `tender_no` ← row["tender_no."]
    `description` ← row["tender_description"]
    `agency` ← row["agency"]
    `award_date` ← row["award_date"]
    `status` ← row["tender_detail_status"]
    `supplier` ← row["supplier_name"]
    `amount` ← row["awarded_amt"]
    `main_category` ← row["main_category"]
Define multiple variant templates for:
    Introduction: include agency, category, tender number, and description
    Award Outcome: vary based on status and supplier
    Status Sentence: express current tender status
if `status` indicates no award **then**
    Select random template from negative award outcomes
else if `supplier` is unknown **then**
    Use predefined sentence for unknown supplier
else
    Select random template from awarded supplier variants
end if
Randomly select:
    one intro sentence
    one award sentence
    one status sentence
Shuffle the three selected sentences randomly
Concatenate into final unstructured text
return final text