CS263, Spring 2023
Midterm Project: Event Argument Extraction
Due: May 30, 2023

Submission Instructions

- Please first fill out this Google Form - https://forms.gle/SsCRr2hT7SMtfjELA to obtain data for the midterm project. You will obtain your unique set of data for this assignment based on the information you provide us. Please submit to receive a copy of the link to the data for future use.
- You will be asked to generate annotation files and have to complete a small coding function for the midterm project on a Google Colab notebook - https://colab.research.google.com/drive/1BQr3j_qUQYqyXK9Kr9lg_lnnxSzmQcyv?usp=sharing. Please make a copy on your drive before editing the notebook. This notebook also comprises functions to check if your file formats are correct.
- Please upload the annotation files and the google colab notebook using the following Google Form - https://forms.gle/gBFdqgUJ3jyEBe9m9. Please make sure the format of the files are correct (using the functions provided on Colab notebook) before submitting them to BruinLearn. You will obtain a confirmation number, and you must include the confirmation number in your final report.
- In the same Google form, we will ask whether if you would like your annotations to be included in our research study. Note that this selection will not affect your midterm project grade.
- Finally, you have to create a final report for the midterm project - report.pdf. You will find the LaTeX template at BruinLearn. For each of the sections, add your write-ups to this report.
- If you find any part of this midterm project unclear, please ask questions in Piazza. Note that parts of this midterm project **do not** have standard answers. The grading is based on the quality of the ontology design, the annotations, and the discussion.

Overview

In this midterm project, we will work on the NLP task of **Event Argument Extraction (EAE)**. EAE falls in the field of Information Extraction (IE) and aims at extracting structured information of event-specific arguments and their roles for events from a pre-defined ontology. This task has various downstream applications like knowledge-graph construction, question-answering, event tracking, summarization and others. In this midterm project, we are interested in **events related to pandemic prediction** and we will guide you to build an event argument extractor for analyzing COVID-related tweets.

As part of the midterm project, we will work on five major components:

- Build a taxonomy for pandemic-related events.
- Annotate a small portion of actual data based on the created taxonomy.
- Design prompt to use ChatGPT to automatically generate EAE labels following your taxonomy.
- Evaluate the ChatGPT predictions with the gold annotations you created.
- Creating input examples that would break ChatGPT / cause it to perform incorrectly.

1 Building an Event Argument Ontology

Events can be described as something that happens or as a state/report of some entity [5, 4]. **Event Triggers** are words that best express the occurrence of an event in a sentence. **Event arguments** are defined as participants in the event which provide specific and salient information about the event. **Event argument role** is the semantic category of the information the event argument provides. We provide an illustration in Figure 1 describing an event about *"Destroying"*, where the event trigger is *obliterated*, and the event consists of argument roles — *Cause*[1] and *Patient*[2].

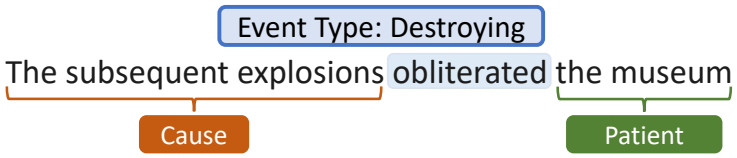

Figure 1: An illustration of the task of Event Argument Extraction (EAE) for the *Destroying* event comprising argument roles of *Cause* and *Patient*. The trigger *obliterated* is highlighted in blue.

In this component, we would like to construct an event argument ontology for extracting events from COVID-related tweets.

You are expected to complete the following partially complete event ontology. More specifically, you have to **add argument roles to events** in the ontology. The partial ontology along with event definitions are described in Table 1 below. We also provide some examples and highlight the event trigger word in **bold**. You can use the event data (obtained from google form above) to understand the examples in order to help to construct the argument ontology.

You can refer to previous EAE ontologies like ACE [2] (Section 6 - https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf) or RAMS [3] (https://nlp.jhu.edu/rams/) for inspiration (but don't merely copy). Some examples of argument roles for events from ontology in different domains are shown in Table 2.

For the submission of this component, we require the following files and write-ups:

Files:

[1] a person or thing that gives rise to an action, phenomenon, or condition.
[2] the semantic role of a noun phrase denoting something that is affected or acted upon by the action of a verb.

Event Name	Event Definition	Sample Examples of events in sentences	Argument Roles
infect	The process of a disease/pathogen invading host(s)	I **caught** a cold ... This disease is **infectious** ... New COVID **cases** ...	You have to complete
spread	The process of a disease spreading/prevailing massively at a large scale	Diseases **sweeps** across L.A. To stop the **spread** of infection ...	You have to complete
symptom	Individuals displaying physiological features indicating the abnormality of organisms	I **got** a fever ... I feel **sick** today ... He **vomits** and coughs ...	You have to complete
prevent	Individuals trying to prevent the infection of a disease	tips to **avoid** catching a cold **Prevention** of infection ...	You have to complete
control	Collective efforts trying to impede the spread of a pandemic	let's **fight** against COVID ... government **measures** to slow down pandemic...	You have to complete
cure	Stopping infection and relieving individuals from infections/symptoms	get **rid** of disease ... how to **cure** this disease... **treatment** for COVID ...	You have to complete
death	End of life of individuals due to infectious disease	**fatality** rate is high ... **loss** of life by COVID ... 1000+ **dead** due to pandemic	You have to complete

Table 1: Partial event ontology for pandemic prediction. There are a total of 7 events listed in the first column along with their descriptions in the second column. Some short examples of these event mentions in actual sentences are shown in the third column. The fourth column needs to be completed as part of the assignment

- ontology.json: This JSON-format file contains your ontology. For each line, you will save a dictionary of the following fields: (1) EVENT_NAME - name of the event, (2) ARGUMENT_ROLE - name of the argument role, (3) ROLE_DESCRIPTION - one line description of the argument role, and (4) EXAMPLE_SENTENCE - one hypothetical example of how the argument can appear in a sentence, use [ ] to tag the argument role in the sentence. List only one argument role per line. For multiple roles of an event, duplicate the event name column but list them in separate lines. You can download the reference argument ontology here.

Write-up: Describe the process by which you created the ontology. Put in any comments/considerations about your ontology.

2 Creating a Gold Annotation

After designing an EAE ontology in the first component, the next step is to annotate data based on this ontology. We will provide you 15 actual sentences from Twitter that discuss different aspects of the pandemic. You have to annotate the sentences with your designed argument roles, i.e. Given a sentence and an event, find word phrases in the sentence that corresponds to any argument roles

Event Name	Argument Roles
Be born	Person, Time, Place
Transport	Agent, Artifact, Vehicle, Price, Origin, Destination, Time
Attack	Attacker, Target, Instrument, Time, Place
Verification	Inspector, Unconfirmed content, Medium, Time, Place
Request	Speaker, Medium, Message, Addressee, Time

Table 2: Examples of argument roles for various events from previous EAE ontologies.

Input Sentence	Event Name	Event Trigger	Argument Annotations
Barrack Obama was born in 1961 in Honolulu	Be born	born	Person: Barrack Obama Time: 1961 Place: Honolulu
Goods are being shipped from Los Angeles to Miami via plane	Transport	shipped	Artifact: Goods Vehicle: plane Origin: Los Angeles Destination: Miami
These attacks have become more frequent	Attack	attacks	
Liam's bags were detained for further inspection	Verification	inspection	Unconfirmed content: Liam's bags

Table 3: Examples of argument role annotations for different events.

of the event. It's possible that an argument role is not mapped in the sentence or the sentence may have no arguments at all as well. We provide some examples of this annotation for several events from previous EAE ontologies in Table 3. Note the format of the last column.

You have downloaded the data for this component from the Google Form link earlier. As part of the data, each student has been provided with a unique set of sentences along with the identified event + triggers. Based on your design of the ontology, you need to annotate these sentences for the specific events. There are two files in the data - 'in_context.csv' and 'eval_data.csv'. The use-case of the two files will be explained in more detail in Section 3. For this part, you have to annotate both data files. Do not use ChatGPT/LLMs for creating your annotations for the data. This should be solely done based on your best understanding.

For the submission of this component, we require the following files and write-ups:

Files:

- in_context-annotated.json: This JSON-format file contains a list of your gold annotations for 'in_context.json' file. For each datapoint, you will save a dictionary of the following fields:

(1) INPUT_TEXT - input sentence, (2) EVENT_NAME - name of the event, (3) EVENT_TRIGGER - the trigger word in the sentence, (4) ARGUMENTS - python dictionary of argument roles mapped to corresponding annotated arguments in the input sentence.

- eval_data-annotated.json: This JSON-format file contains a list of your gold annotations for 'eval_data.json' file. For each datapoint, you will save a dictionary of the following fields: (1) INPUT_TEXT - input sentence, (2) EVENT_NAME - name of the event, (3) EVENT_TRIGGER - the trigger word in the sentence, (4) ARGUMENTS - python dictionary of argument roles mapped to corresponding annotated arguments in the input sentence.

Write-up: For the write-up, please discuss your process of annotation. Describe ambiguous cases if any.

3 Model Prediction on your data

Next, we want to evaluate the performance of models on this small dataset. More specifically, we are interested in evaluating the zero-shot ability of Large Language Models (LLMs) like ChatGPT on this mini-data.

To start with, create an account and get access to ChatGPT here - https://chat.openai.com/. **ChatGPT may be overbooked/overused at times, so do attempt this step early.**

To use ChatGPT for prediction on this task, we need to have the following considerations:

Prompt Engineering We want to utilize ChatGPT to make extract arguments from the input sentences provided in the data (Section 2). You can only utilize the event information and argument role name for argument extraction from the sentences. Since ChatGPT only takes text inputs, the challenge is to remodel the task of EAE into a text-to-text form and **generate an appropriate prompt** for the same. Some examples of how to create prompts for some other NLP tasks have been provided in Table 4 below. You might want to think about how to write a good prompt for the task of EAE.

Task	Sample Prompt
Sentiment Classification	What's the sentiment for the sentence: "{Text Input}", positive, negative, or neutral?
Translation	Translate English to French: {Input}
	Translate the following phrase from English to French: {Input}
Q & A	{Text Input} Question: {Question Input}. True, False, or neither?
Paraphrase Generation	Generate a paraphrase of the following sentence: {Text Input}.

Table 4: Examples of prompts for NLP tasks. [1, 6]

In-context Examples LLMs also perform better with in-context examples. In-context examples are 1-5 examples of the task that are provided as part of the prompt that help the LLMs to

Figure 2: An illustration of the In-context learning in LLM

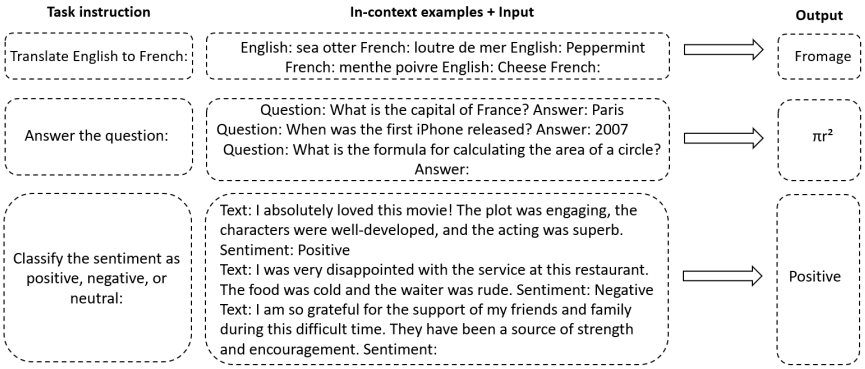

Figure 3: ICL examples for machine translations, question answering, and sentiment analysis

understand the task better and learn what's expected to be the ideal output. An illustration is shown in Figure 2. Some examples of the usage of in-context examples on other NLP tasks are shown in Figure 3 below. You might want to think about how to use in-context examples for the task of EAE.

Output Consistency Since ChatGPT is a generative model, we have less control on the output format i.e. there is no fixed format/structure in which ChatGPT generates outputs. But since we want to extract precise information about the arguments, we want the output of ChatGPT to follow a pattern/fixed structure (also called output mapping). In Table 5, we show examples of consistent and inconsistent behaviors of ChatGPT outputs.

(a) For the first task of sentiment analysis, we want the outputs to be one of the three: positive/negative/neutral so that we can use the mapping. But if the model generates anything else, the output is inconsistent, as we can't automatically map the output to one of the three labels.

(b) Second, we take an example of the question-answering task. We decide for an identity output mapping such that our output can be directly considered as the answer i.e. whatever the model produces is our final answer. But if the model generates natural language sentences, like shown in the examples, then the output is inconsistent to the mapping.

You might want to experiment with various prompts for EAE such that the outputs of ChatGPT are consistent to a single output mapping.

Based on the three considerations above, we want you to explore writing various kinds of prompts (from very simple to complex) to extract arguments from the corresponding sentences. You will be graded based on the variety and extent of various prompts you explore.

More specifically,

Task	Sample Input	Output Mapping	Model Output	Behavior
Sentiment Classification	What's the sentiment for the sentence: {sent}, positive, negative, or neutral?	positive = 1 negative = -1 neutral = 0	negative positive neutral	Consistent
			slightly positive ambiguous yes	Inconsistent
Question Answering	George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat?	{identity}	dry skin wet skin rough skin	Consistent
			the skin should be dry skin that is wet it seems like the skin should be rough	Inconsistent

Table 5: Examples of Consistent and Inconsistent Behaviors of Language Models. [1]

(a) you have to tune your prompts ONLY using the sentences from the 'in_context.json' file. Do not try prompts on sentences from the 'eval_data.json'. Based on your best judgement, choose the best prompt. Do not choose the best prompt based on evaluation metrics.

(b) you need to fill a small python function 'argument_extractor' (in the google colab) as the output mapping to extract the predicted arguments given the ChatGPT output text and the corresponding roles. This will be done only for your best prompt.

(c) you have to use your best prompt on the 'eval_data' to get the model outputs. Then use the 'argument-extractor' script to extract the model predictions. You can use the sentences from 'in_context.json' for adding in-context examples to your prompts.

For the submission of this component, we require the following files and write-ups:

Files:

- logs.json: This file contains the logs of all your experiments with ChatGPT on the sentences ONLY from the in-context file. Make a note for each prompt you tried and corresponding ChatGPT outputs in this file. Each sentence in this file should comprise four fields - (1) INPUT_TEXT - actual input sentence, (2) PROMPT - actual prompt provided to ChatGPT (along with the input sentence), (3) OUTPUT_TEXT - text output from ChatGPT based on the prompt, (4) EXTRACTED_ARGUMENTS - python dictionary of argument roles mapped to corresponding extracted arguments from the output text.
- pred.json: The second file contains the input/output from ChatGPT only for the best prompt for all the sentences from the eval-data. Each sentence in this file should comprise four fields for each sentence from the eval-data - (1) INPUT_TEXT - actual input sentence, (2) PROMPT - actual prompt provided to ChatGPT (along with the input sentence), (3) OUTPUT_TEXT - text output from ChatGPT based on the prompt, (4) EXTRACTED_ARGUMENTS - python dictionary of argument roles mapped to corresponding extracted arguments from the output text.

Write-up: We want you to write about your observations of using different kinds of prompts and corresponding ChatGPT outputs. We also want you to discuss how did you choose the best kind of prompt.

4 Evaluation of model

The final component includes using your created gold annotations (Section 2) to evaluate the extracted argument predictions of ChatGPT (Section 3). For evaluation, we will report overall micro precision, recall, and F1 score.

In order to compute each of these metrics, we will do an exact match of the gold argument with the best-predicted argument. If a single word/character doesn't match, we treat it as a miss. We provide an example of the same in Table 6.

Input Sentence	Gold Argument Annotations	Model Argument Annotations	Eval Metrics
Barrack Obama was born in 1961 in Honolulu	Person: Barrack Obama Time: 1961 Place: Honolulu	Person: Barrack Time: 1961	Prec: 50% Rec: 33% F1: 40%
Goods are being shipped from Los Angeles to Miami via plane	Artifact: Goods Vehicle: plane Origin: Los Angeles Destination: Miami	Artifact: Goods Artifact: aeroplane Origin: Miami Agent: Los Angeles	Prec: 25% Rec: 25% F1: 25%
These attacks have become more frequent		Time: frequent	Prec: 0% Rec: - F1: -
Overall			Prec: 2/7 Rec: 2/7 F1: 2/7

Table 6: Examples of argument role annotations for different events.

For the submission of this component, we just need a write-up:

Write-up: Report the final precision, recall, and F1 score for the ChatGPT extracted arguments. Also, provide a qualitative analysis for the kind of mistakes ChatGPT makes and potential ways to improve that mistake.

5 Breaking ChatGPT

This is a fun exploratory part and the open-ended part of the assignment. We would like you to design EAE examples/data that would break ChatGPT or cause ChatGPT to make errors. Please restrict yourself to the task of EAE only and do not report examples for other kinds of tasks. Some examples of doing so include:

- **Adversarial Examples:** You can generate adversarial examples by performing minimal edits/paraphrasing the input sentence and causing the ChatGPT model to perform worse than before.
- **Hallucination Examples:** You can design underspecified sentences such that the ChatGPT models tries to hallucinate based on its previous knowledge.
- **Fairness Perspective:** You can evaluate if ChatGPT changes model outputs based on the identity of the people involved in the input.
- **Create your own examples:** You can be creative and create your own tricky examples involving these events on which ChatGPT fails to perform well.

For the submission of this component, we require the following files and write-ups:

Files:

- break-gpt.json: This file comprises all examples that you came up with on which ChatGPT doesn't perform well. Provide at least 10 cases, but you're encouraged to provide more cases you found or discuss a systematic way to generate these examples. Each sentence in this file should comprise five fields for each sentence from the eval-data - (1) INPUT_TEXT - actual input sentence, (2) EVENT_NAME - name of the event in the input sentence, (3) EVENT_TRIGGER - trigger of the event in the input sentence, (4) PROMPT - actual prompt provided to ChatGPT (along with the input sentence), (5) OUTPUT_TEXT - text output from ChatGPT based on the prompt, (6) EXTRACTED_ARGUMENTS - python dictionary of argument roles mapped to corresponding extracted arguments from the output text. (7) EXPECTED_ARGUMENTS - python dictionary of argument roles mapped to actual expected arguments from the output text.

Write-up: Provide your experience and way of approaching how to come up with examples to break ChatGPT.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle,

M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[2] George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, pages 837–840. Lisbon, 2004.

[3] Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. Multi-sentence argument linking. *arXiv preprint arXiv:1911.03766*, 2019.

[4] Rujun Han, I-Hung Hsu, Jiao Sun, Julia Baylon, Qiang Ning, Dan Roth, and Nanyun Peng. Ester: A machine reading comprehension dataset for reasoning about event semantic relations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7543–7559, 2021.

[5] James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. The timebank corpus. In *Corpus linguistics*, volume 2003, page 40. Lancaster, UK., 2003.

[6] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.