Translating English Messages into DAIDE using LLMs with Templates

1 Introduction

The aim of this project is to develop a more accurate method to convert natural language into DAIDE that can be used as part of a Diplomacy bot, using large language models to perform this translation. The use of templates filled by GPT-4 has seen an accuracy of 30%, compared to the current accuracy of 15% obtained from converting English into AMR and AMR into DAIDE.

2 Dataset

The dataset consists of 312 instances of English messages and their corresponding DAIDE translations, found here: https://github.com/ALLAN-DIP/diplomacy_translation/blob/main/data/annotated_daide.json. This dataset was generated through converting gold AMR to DAIDE, and presumably taking all the translations that pass the `daidepp` parser.

3 Method

Templates are a set of DAIDE messages with specific power and province names replaced with placeholders (POWER and PROVINCE respectively), leaving a DAIDE snippet that can be filled in by the LLM. For instance, a template for a DAIDE message `PRP (DMZ (AUS RUS) (GAL))` is `PRP (DMZ (POWER POWER) (PROVINCE))`. In this manner, a template retains the structure of the DAIDE. These templates were learned from the training set, where templates given to the model were those that occurred more than once in the set.

OpenAI's GPT-4 was prompted with an intial DAIDE overview prompt, followed by six examples for in-context learning, referred to as the "initial prompt".

> There is a language called DAIDE that is used to communicate orders in Diplomacy.
> All DAIDE tokens are three uppercase letters.
> Here is part of the abstract syntax tree for DAIDE:
> press_message = PRP (arrangement)
> arrangement = PCE (power power+)
> arrangement = ALY (power power+) VSS (power+)
> arrangement = XDO (order)
> arrangement = DMZ (power+) (province+)
> arrangement = AND (arrangement)+
> arrangement = SCD (power centre+)+
> reply = REJ/YES (press_message)
> order = unit MTO province
> order = unit BLD
> unit = (power type province)
> Here are some translation examples from English to DAIDE:
> [England to Russia] I'm moving Edi into NTH, and NTH into helgo
> AND (XDO ((ENG FLT EDI) MTO NTH)) (XDO ((ENG FLT NTH) MTO HEL))
> [Germany to England] I'd like denmark, norways all yours. Perhaps we could move on Russia but I think we should start with france, otherwise he'll become a problem for us both. After that we can hash out the rest of scandanavia and Russia as it goes?
> PRP ((SCD (GER DEN)) (SCD (ENG NWY)) (ALY (GER ENG) VSS (FRA)))
> [England to Germany] Sorry about the delay. I'm willing to exchange Belgium for cooperation against France.
> YES (PRP (AND (SCD (ENG BEL)) (ALY (ENG GER) VSS (FRA))))
> [Russia to England] Can your army in Warsaw support my army in Ukraine? PRP ((ENG AMY WAR) SUP (RUS AMY UKR))
> [Germany to Austria] Can your fleet on the Baltic Sea support my army in Sweden?
> PRP ((AUS FLT BAL) SUP (GER AMY SWE))

[France to Italy] Can your fleet in the Adriatic Sea convoy my army in Apulla to Trieste?
PRP ((ITA FLT ADR) CVY (FRA AMY APU) CTO TRI)

Translations were generated in a two-part process, with 2 queries to the LLM per sentence to translate. For the first query, the initial prompt was given, along with a list of generated templates, which were generated by taking all templates that occurred more than once in the given training set. The model was then asked to choose the DAIDE template that best represented the given English message. The specification to give only the DAIDE template was included due to the verbosity of GPT-4 otherwise.

Here are some DAIDE templates:
PRP (DMZ (POWER POWER) (PROVINCE))
[...]
Choose the best DAIDE template to represent this message: <msg>.
Give only the DAIDE template.

The template obtained from the first query was used to obtain the translation in the second query. The initial prompt was given again, and this time the model was given the template and asked to translate the message by replacing the placeholders POWER and PROVINCE:

Use the given DAIDE template and replace POWER and PROVINCE accordingly to translate this message: <msg>
DAIDE template: <template>
Give only the DAIDE message.

4 Results

See the table below for selected results.

In terms of the accuracy metrics, string equality considers only if the two DAIDE strings are equal, with string equality considers if the DAIDE strings are equal while accounting for possible order differences for both tokens and arrangements (e.g. for PCE, the powers can be listed in any order). The original F-score was the metric by which a previous researcher calculated similarity, and was included for comparison. However, its main issue is that it overstates the accuracy by only calculating the overlap in tokens between strings, so DAIDE orders such as `PRP (ALY (TUR AUS) VSS (RUS))` and `PRP (ALY (TUR RUS) VSS (AUS))` are equivalent. The better F-score converts the DAIDE strings into trees, generates a list of all parent-children subtrees (the parent node and its immediate children), and then computes an F-Score based on the overlap, accounting for order differences.

For the different approaches, the first row details the current approach, the second row was generated by trying to directly translate from English to DAIDE using the above prompt, and the third two was created by using GPT-4 to select among 4 possible DAIDEs which were generated by modifying the prompt. Note that these 3 results were obtained using a random sample of 100 instances from the larger dataset.

Method	String equality	String equality without order	Original F-Score	Better F-Score
Current ENG → AMR → DAIDE	0.130	0.150	0.553	0.459
GPT-4 with original prompt and [speaker to speaker] tag	0.100	0.100	0.707	0.547
Selecting best option among 4 previously generated DAIDEs	0.140	0.140	0.598	0.459
Results from template cross-validation	0.281	0.299	0.801	0.672

Table 1: Comparison of different approaches with a variety of accuracy metrics

Initial results show a doubling in performance over the baseline English to AMR to DAIDE model of 15% exact match (string equality without order) to 30%.

Results were obtained using 10-fold cross-validation over the entire dataset, with templates generated on the train portion, and translations performed on the test portion. Results were consistent across folds, with an average of 29.9% exact match, a minimum of 28.1% and a maximum of 33.7%.

In terms of the templates generated, on average 25 templates were generated on each of the training sets, and covered an average of 62.7% of the instances. In other words, 62.7% of the instances in each training set were represented by one of the generated templates.

An interesting thing to note during the first query is that sometimes the LLM returns a filled-in template, rather than the generic template with placeholders.

For some messages, the LLM is also able to identify that none of the given templates fit, most visibly in cases where the English message is meaningless out of context, e.g. a message consisting of just "Bud".

This method outperforms other LLM translation methods I have attempted, such as directly asking the LLM to translate from English to DAIDE, selecting the best option from among 4 previously generated DAIDEs (as seen in Table 1), or trying to convert into AMR and AMR into DAIDE. This is likely to due to the relative ease of the task - picking out one template, and filling in a few placeholders. Applications outside DAIDE is limited due to the restricted nature of DAIDE syntax, but could be an interesting avenue to explore.