LLM-Powered Tool for Automatically Generating Fictional World Maps

Teni Asade

CPSC 4900

Advisor: James Glenn

Background

As readers of Tolkien, George R. R. Martin, or Sarah J. Maas might know, fictional fantasy worlds are often vast, mirroring the complexity of our own geography. The aspiring fantasy writer has to manage a complex group of places and routes while also implementing character development, vivid descriptions, and gripping plots.

Problem Description

There are many tools to help writers map out their complex worlds such as World Anvil and Campfire. Aeon Timeline can help manage the order and length of events. However, all of these tools stand alone and require the author to divert time and attention away from their manuscript to craft something new. I propose a tool that periodically ingests the writer's text *as* they write, using Natural Language Processing and constraint satisfaction to generate a map depicting the author's world as they create it on the page. This passive, in-editor map generator will convert narrative text into spatial constraints and flag inconsistencies between said constraints without the writer having to create and check a map on some external site, allowing them to actually focus on what matters most: writing.

Main goal

The goal of this project is to create a Google Docs plug-in that will periodically sample and ingest lines of text describing movement e.g. ‘We travelled north from Rivenfell to Callahan on foot in 30 days’ and output a map with the relevant places and correctly modelled distances and locations – in this case, a map that has Rivenfell and Callahan and some indication that they are at most 900km apart with Callahan’s latitude correctly representing that it is north of Rivenfell.

Approach

My plan is to start with a minimum viable product (MVP) that correctly generates a simple map (dots on a plane with coordinates) for a simple input like the one above. This will involve:

1. Named Entity Recognition and relation extraction
2. Constraint extraction
3. Constraint solving using least squares and flagging unsolvable constraints using a constraint satisfaction solver in the background to check if any subset of constraints are strictly contradictory
4. Map rendering

This MVP will be built with assumptions including the assumption that we are constructing a 2D Euclidean map using default speeds and straight-lines. Once this MVP has been created I will begin adding other important functionality:

1. Ensuring support for multiple places
    1. Three-plus-way relation e.g. “Rivenfell to Callahan to Erendale”

    2. Multiple places but not all related e.g. Rivenfell to Callahan then Erendale to Camelot.
2. Ensuring support for different modes of transport
3. Conducting experiments to increase the correctness of the generated map with regards to more complex input documents

Once the tool has met these extra requirements satisfactorily, I will integrate it as a Google Docs plug in. This will include a simple custom UI that allows the author to override set walking/sailing/etc speeds and potentially even include new modes of transportation e.g. flying by pegasus. The plug-in will also visually flag lines of text in the document that lead to unsolvable constraints. I will also improve the way the map is visually rendered at this stage.

I will continue fine-tuning the tool, mostly in terms of the inferences that the LLM makes by:

1. Integrating additional support for context e.g. instead of assuming a hard-coded default mode of transportation of going by foot when no mode of transportation is explicitly stated, the tool will use the context of the story to make a more informed assumption about the mode of transportation used. The LLM could also infer relations between places where no explicit relation is stated.
2. Ensuring support for edge cases e.g. “The group camped halfway”. This also includes taking into consideration the route the party used or didn’t use and using this to infer other things about the geography of the world. For example, if the party completed the journey in a longer time frame than expected and somewhere else in the text it states that they needed to avoid mountains without explicitly mentioning where these mountains

were the LLM might infer the location of the mountains and place them on the map with a UI element showing that it is an inference.

All inferences will be marked visually on the generated map.

Tools

1. **Frontend:** React + Leaflet
2. **Google Docs Integration:** Google Apps Script
3. **Backend:**
    - Python (SpaCy/Stanza + OR-Tools)
    - LLM such as OpenAI GPT-4, LLaMA 3 or Mistral

Research Component/Anticipated Challenges

The research component of this project will involve fine-tuning the Named Entity Recognition, especially considering that many of the fantasy place name keywords will be unique. The solution might involve the user inputting their place names but the goal is to avoid the requirement of such interruptions to the writing process. In general, the unique element of this project – and what I expect to be most challenging – is that the data being used is fictional and so I am excited to learn about and improve the LLMs interactions with unique, fictional data. Another exciting challenge I anticipate is that of the LLM using assumptions to flesh out the map. I will have to balance the benefit of the LLMs correctly inferring information about the

world with the probability of incorrect assumptions and subsequent disruptions to the writer as they correct said incorrect assumptions. I look forward to tackling these challenges.

Tentative Deliverable Schedule

DELIVERABLE	DUE DATE	DURATION
Proposal Presentation	*Sept 19th*	-
Set up backend tech NER and travel constraint extraction for simple input	Sept 21st	2 weeks
Constraint solving and map rendering for simple input including flagging unsolvable constraints	Oct 5th	2 weeks
Progress Report	*Oct 10th*	-
Support for multiple places and modes of transport (NER and travel constraint extraction for more complex input)	Oct 26th	3 weeks
Integrate tool as a Google Docs plug in - add UI that allows the user to configure the tool Improve visual rendering of the map	Nov 9th	2 weeks
Progress Presentation	*Nov 14th*	-
Integrate tool as a Google Docs plug in - input is now fed from Google Docs to tool	Nov 23rd	2 weeks
Final Report (to advisor)	*Dec 1st*	-
Final Poster	*Dec 2nd*	-
Further fine-tuning	Dec 7th	2 weeks
Edited Final Report	*Dec 11th*	-