Fantasy Story Map Builder

A Procedural Tool for Automatically Generating Fictional World Maps

Teni Asade
teni.asade@yale.edu

Advisor: James Glenn
james.glenn@yale.edu

A Senior Thesis as a partial fulfillment of requirements
for the Bachelor of Science in Computer Science
Department of Computer Science
Yale University
Nov 29, 2025

Acknowledgements

I want to express my gratitude to my advisor James Glenn who gave me the room I needed to explore this idea. I would also like to express my gratitude to Sohee Park for leading the course. I am also grateful to the Yale Department of Computer Science for the skills and knowledge, as well as friends and great memories, that it has provided for me.

Fantasy Story Map Builder: A Procedural Tool for Automatically Generating Fictional World Maps

Teni Asade

Abstract

This project presents *Story Mapper*, a procedural tool that automatically generates fictional world maps directly from narrative prose using Natural Language Processing (NLP) and constraint satisfaction. Fantasy writers often describe spatial relationships implicitly through references to movement, direction, and travel time, but converting these descriptions into coherent maps is typically a time-consuming, manual process. *Story Mapper* integrates map generation into the writing environment itself through a Google Docs plug-in, allowing authors to visualize their worlds as they create them on the page.

The system extracts spatial information from user-selected text using named entity recognition, dependency parsing, and relation extraction. Travel durations are converted into approximate distances, enabling both qualitative cues (e.g., *north of*) and quantitative cues (e.g., *30 days on foot*) to be expressed as spatial constraints. Contradictions, such as impossible directions or conflicting distances, are stored. The non-contradictory constraints form a Constraint Satisfaction Problem in which each location is represented as a variable with unknown coordinates. A solver implemented in Python computes a spatial layout that satisfies the maximum number of constraints.

A visualization module then renders the resulting layout as a terrain-style map using procedural generation techniques and displays it within the Google Doc. Conflicts are highlighted both on the map and in the text, offering immediate, actionable feedback without interrupting the writing process.

Overall, *Story Mapper* demonstrates how narrative language can be translated into structured spatial representations. It provides a practical authoring aid for maintaining world coherence and contributes to broader research on computational creativity by showing how AI-driven tools can augment, rather than replace, human storytelling.

1 Introduction

Eighty-one percent of Americans say that they want to write a book. Only fifteen percent ever begin, and of that small fraction, just three percent finish. Writers face numerous barriers – perfectionism, limited time, self-doubt, and structural challenges like weak plots. Fantasy writers in particular face all these hurdles and more. They must construct entire worlds, managing vast groups of places and routes that mirror the complexity and intricacy of our own geography.

Although there are many digital tools for fantasy cartography, most exist outside the writing process. Authors must leave their manuscript, open a separate interface, and manually design maps – an interruption that fragments creative flow and increases cognitive load, making the already difficult process of creating a novel that much harder. This paper asks a simple but significant question: **Can we leverage Natural Language Processing (NLP) and constraint satisfaction to automatically generate maps from narrative text as authors write?**

This research situates itself at the intersection of computational creativity, NLP, and constraint satisfaction. Prior work has explored textual world-building, story understanding, and spatial reasoning independently, but little connects these to the author's live creative workflow. My contribution is a tool – *Story Mapper* – that ingests text from a writer's document, identifies phrases describing movement or location (e.g. "We traveled north from Rivenfell to Callahan in thirty days"), and converts them into spatial constraints. The system uses these constraints to generate an evolving map and flags inconsistencies, allowing authors to maintain world coherence without having to leave their text editor.

The resulting Google Docs plug-in demonstrates how NLP can be used not only to analyze or summarize stories but to augment the act of storytelling itself. This paper presents the design, implementation, and evaluation of Story Mapper, discusses the NLP and constraint-resolution methods used, and highlights its broader implications for the future of creative technologies and computer-assisted writing.

2 Background

This paper is based on three main pillars: Natural Language Processing, Constraint Satisfaction & Spatial Reasoning, and Computational Creativity & Narrative Technologies.

1. Natural Language Processing

Natural Language Processing (NLP) is the branch of Artificial Intelligence research focused on enabling computers to understand, interpret, and generate human language. In this project, NLP is used to identify and extract references to locations, directions, and distances in fictional text. Techniques used include **Named Entity Recognition** to identify place names such as *Callahan* and *Rivenfell* in the input "The party travelled north from Callahan to Rivenfell in thirty days", **dependency parsing** to identify how words depend on one another. In the previous example, *Rivenfell* depends on *from* and every word in the sentence directly or indirectly depends on the main verb *traveled*. **Relation extraction** is used to find semantic relationships between entities. For example, the pair *(Rivenfell,Callahan)* is connected by the relationship *north of* or *30 days apart*. Modern NLP models, particularly large language models, improve context understanding and offer new possibilities for interpreting creative or metaphorical descriptions that would challenge older, rule-based parsers.

2. Constraint Satisfaction & Spatial Reasoning

Spatial reasoning involves understanding how objects and places relate in space, both qualitatively (e.g. *north of*) and quantitatively (e.g. *100km apart*). Constraint satisfaction provides a mathematical framework for representing and solving such relationships. A Constraint Satisfaction Problem (CSP) is a set of variables with possible values that must satisfy certain constraints. In the example of Story Mapper, the variables are locations, their possible values are coordinates, and the constraints that they must satisfy are the distances and directions specified in the text. By modeling locations as variables and narrative descriptions as constraints and then assigning values to the various constraints such that all constraints in the system are satisfied, the system can infer a consistent spatial layout of the fictional world and generate a valid map. If the constraints are unable to be satisfied then that means that there is an impossibility or contradiction in the text, which is then flagged to the author.

3. Computational Creativity & Narrative Technologies

Computational Creativity is a subfield of artificial intelligence concerned with the application of computer systems to simulate or enhance human creativity. Within narrative domains, this has led to systems that generate stories, visualize plot structures, or suggest character arcs. This project contributes to the latter objective of creative computing: to design systems that support human creativity as opposed to mimicking it. It does so by helping authors maintain spatial coherence in their creative writing. It also contributes to research in narrative intelligence, the subfield of AI focused on understanding and generating stories.

4. The Creative Writing Process

A staple of many fantasy novels is intricate, large-scale worldbuilding. This has led to the creation of numerous tools for manual map-making – Inkarnate, Wonderdraft, World Anvil, and more. As noted, none of these tools are integrated with text-editors and all require the author to manually create the map. This tool aims to automate the process, not to take away from the author's creativity but to allow them to devote time to generating those creative ideas on the page. *Story Mapper* is simply a tool that visually maps out what authors have already done the hard and beautiful work of generating.

3 Methodology

The system consists of two primary components: a back-end program for map generation written in Python and a Google Docs integration for text ingestion written in Google Apps Script. Figure 3.1 below illustrates the overall architecture of the tool.

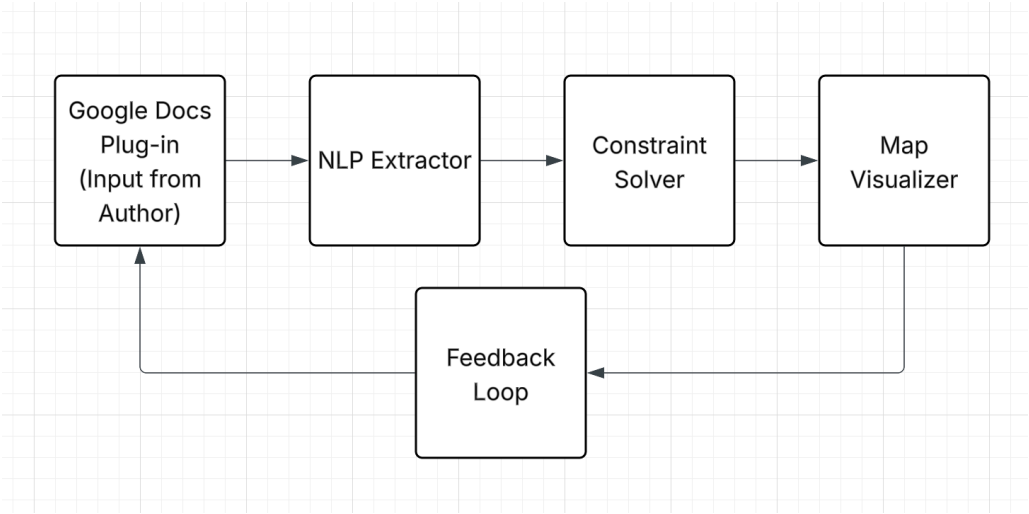

Figure 3.1

System Workflow

As the author writes in Google Docs, the plug-in samples the text and sends the sentences to the back-end. The back-end performs Natural Language Processing (NLP) to extract spatial information, converts it into structured constraints, and creates a constraint-satisfaction model representing the world. The visualizer then renders a map, which is embedded within the Google Doc for the author to view. The workflow can be summarized as:

Author text → Text extractor → Constraint Solver → Map visualizer → Feedback to Author

1. Extractor Module

The extractor parses narrative sentences to identify key entities such as locations, directions, and distances. It uses spaCy for dependency parsing and named entity recognition, regular expressions for converting descriptions of time into number of days (e.g. *three weeks* into *21*), as well as the word2number Python library to convert written numerals into numeric form (e.g. *five* to *5*). When temporal data are encountered (e.g. *30 days on foot*), the system uses a calibrated average travel speed to estimate corresponding spatial distances (e.g. *600 km*). Inconsistencies (conflicting distances or directions) are found and stored.

2. Solver Module
The solver formulates a constraint satisfaction problem (CSP) in which each location is a variable with unknown coordinates. The extracted relationships (*north of*, *600 km apart*, etc.) are modeled as constraints. Using NumPy and SciPy, the solver computes a spatial configuration that satisfies the maximum number of constraints, and updates the map accordingly.

3. Visualizer Module
The visualizer, implemented using Matplotlib, transforms the solver's coordinates into a visual terrain map. It first normalizes all points to fit within a fixed image size with padding. Using NumPy, the renderer constructs a heightmap-like terrain by placing Gaussian bumps centered at each normalized location and then normalizing and squaring the resulting field to ensure low-elevation "ocean" regions around the periphery. Procedural noise (Perlin) is added globally to introduce natural variation and avoid unnaturally smooth contours. The module then overlays the location labels, pairwise routes with annotated distances, and visual warnings for flagged conflicts (either distance or directional). The final image is rendered with Matplotlib, styled with a custom colormap, and saved to disk for display in the UI.

Implementation Details
The system is implemented primarily in **Python 3.11,** with integration scripts connecting to the Google Docs API for text retrieval. Dependencies include *NumPy*, *SciPy*, *Matplotlib*, *Transformers*, *spaCy*, *re*, *word2number*, and *collections*. Development and testing were performed on macOS and Ubuntu environments.

4 Results

To analyze the results of Story Mapper, let us walk through a simple example:

"The group travelled from Callahan to Rivenfell on foot in 30 days. Callahan is north of Rivenfell."

4.1 User Interface in Google Docs

The user writes the above input in a Google document, clicks the 'Story Map' menu button, and then selects 'Generate Map'. A request is sent to the back-end through a Google Apps Script. See Appendix A for images of the workflow.

4.2 Extraction Performance

On the backend, the following information is extracted from the input:

--- Sentence 1 ---
Locations: Callahan, Rivenfell
Dates: ['30 days']
Direction: []
Entry: [Sentence 1: The group travelled from Callahan to Rivenfell on foot in 30 days.]

--- Sentence 2 ---
Locations: None
Dates: []
Direction: [('Rivenfell', 'Callahan', 'north')]
Entry: [Sentence 2: Callahan is north of Rivenfell.]

4.3 Constraint Satisfaction and Spatial Modeling

The distance constraint for the input looks like this:

{('Callahan', 'Rivenfell'): [(600.0, [Sentence 1: The group travelled from Callahan to Rivenfell on foot in 30 days.], 'real')], ('Rivenfell', 'Callahan'): [(200, [Sentence 2: Callahan is north of Rivenfell.], 'default')]}

The description of the journey taking *30 days* is converted into a distance of *600km*. This is associated with the locations Callahan and Rivenfell and the sentence *The group travelled from Callahan to Rivenfell on foot in 30 days*. This distance is marked as *real* as it is evidenced by the text.

The sentence *Callahan is north of Rivenfell* also results in a distance constraint, particularly the default distance of *200km*, to allow the solver to place locations in space even when no distance is specified by the text. Given a *real* and *default* distance for a location pair, the *real* distance is always used.

The direction constraint for the input looks like this:

{('Callahan', 'Rivenfell'): [((0, -1), [Sentence 2: Callahan is north of Rivenfell.])]}

These constraints are passed into the solver and result in coordinates such as:

{'Callahan': (np.float64(11040.299910716261), np.float64(1959.0870424244456)), 'Rivenfell': (np.float64(11040.299910716261), np.float64(1759.0870424198831))}

4.4 Map Visualization

A map is generated and sent back to the Google Apps Script that originated the request. The Google Apps Script then displays the map within the Google Docs text-editor.

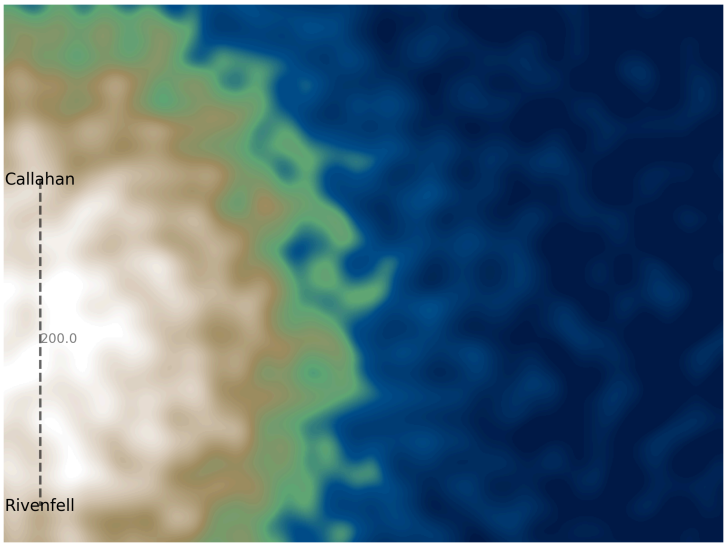

Figure 4.1: Generated map

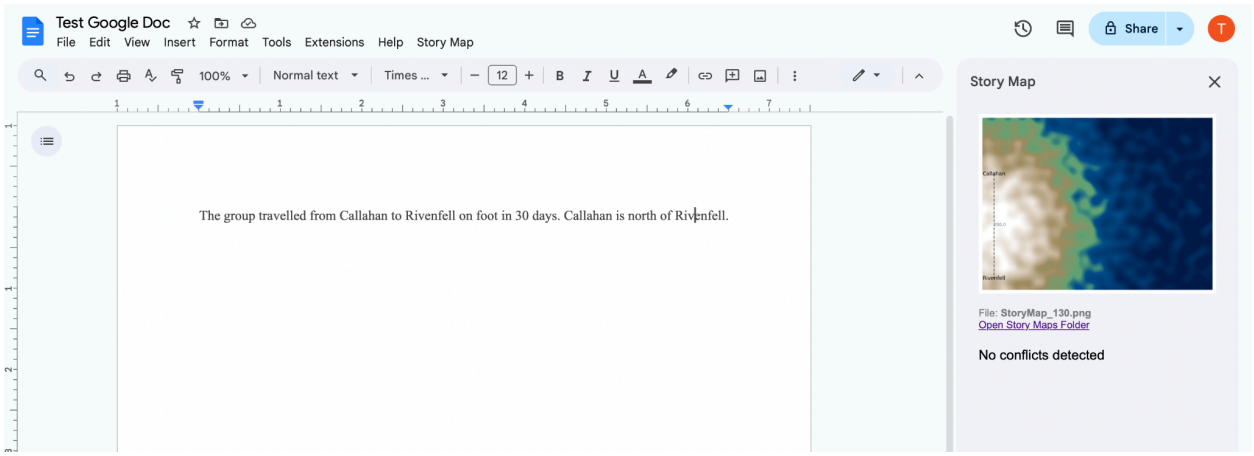

Figure 4.2: Generated map within the context of the text-editor

4.5 Conflict Flagging

Let us consider an input with a conflict:

"Camelot is north of Erendale. Erendale is north of Camelot."

This results in:
Direction: [('Erendale', 'Camelot', 'north')] from sentence 1 and
Direction: [('Camelot', 'Erendale', 'north')] from sentence 2.

The extractor creates direction and distance constraints. It also creates a dictionary of direction and distance conflicts. In this case, as in the case above, the distance conflicts dictionary is empty. However, the direction conflicts dictionary contains:

{('Camelot', 'Erendale'): [((0, -1), [Sentence 1: Camelot is north of Erendale.]), ((0, 1), [Sentence 2: Erendale is north of Camelot.])]}

Whenever there is a conflict of any kind, the first instance of information is stored. That means that, in this case, the first distance constraint – ('Camelot', 'Erendale'): [((0, -1), [Sentence 1: Camelot is north of Erendale.]) – is what is passed to the solver.

Figure 4.3 depicts the resulting displayed map and document. The conflicting sentences are highlighted on the document and also printed below the map. There is also a warning sign beside one of the locations involved in the conflict. Figure 4.4 depicts the same document once a change has been made to rectify the conflict (*Erendale is north of Camelot* is changed to *Erendale is south of Camelot*.)

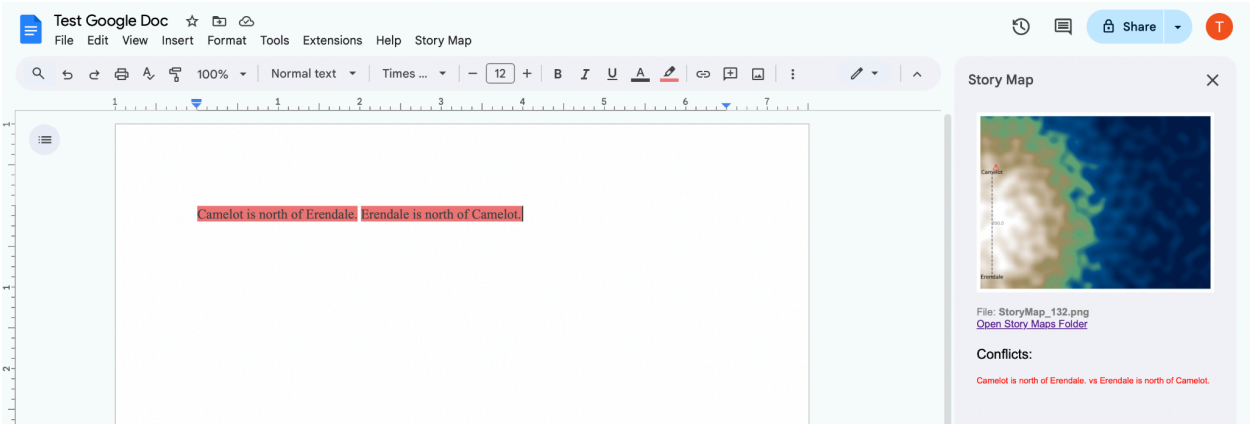

Figure 4.3: Input with one directional conflict.

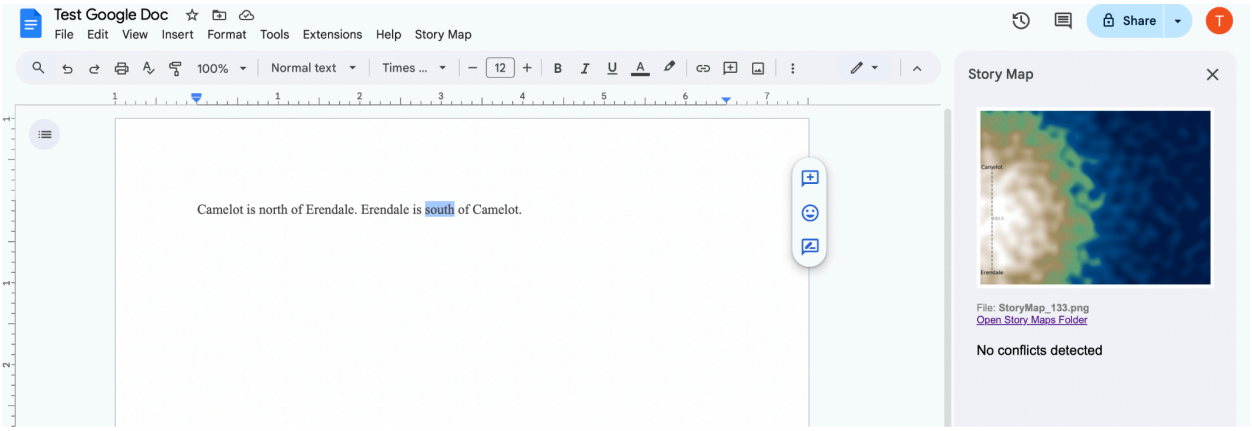

Figure 4.4: Input after conflict is resolved.

4.6 Various Inputs

Example Input	Extracted Locations	Extracted Date	Extracted Distance	Extracted Directions	Significance of Example
The group travelled from Callahan to Rivenfell on foot in 30 days. Callahan is north of Rivenfell (1)	Callahan, Rivenfell	((Callahan, Rivenfell), 30 days)	(Callahan, Rivenfell, 600km)	(Rivenfell, Callahan, north)	Information spread across multiple sentences
He went from Ayo to Lok to Mizani in a week. (2)	Ayo, Lok, Mizani	((Ayo, Lok, Mizani), a week)	(Ayo, Lok, 70 km), (Lok, Mizani, 70km)		Three locations sharing one date/indication of distance

She went north from Troye to Kiv in one day. She went from Troye to Kiv in 30 days. Kiv is south of Troye. (3)	Troye, Kiv	((Troye, Kiv), one day), ((Troye, Kiv, 30 days)	((Troye, Kiv), 20km), ((Troye, Kiv, 600km)	(Troye, Kiv, north), (Troye, Kiv, south)	Input with two conflicts (one directional and one in terms of distance) where one sentence is present in both pairs of conflicts.
When her father travelled north from Rivenfell to Callahan it took him three weeks and for those three weeks Alina would turn into the most sour little girl her small town had ever seen. See, Callahan was southeast of Stonebrooke, the little trading town where Alina's mother had grown up. (4)	Rivenfell, Callahan, Stonebrooke	((Rivenfell, Callahan), three weeks)	(Rivenfell, Callahan, 420km)	(Rivenfell, Callahan, north), (Stonebrooke, Callahan, southeast)	Input where location and travel information is embedded within narrative text

See Appendix B for more detailed information about inputs 2-4 and their results.

These examples demonstrate that the system is able to extract locations, directions, and distance from input. It is able to synthesize data that is spread across multiple sentences and is able to identify conflicts even if the information is split between multiple sentences.

4.6 Summary of Findings

The system can reliably extract and represent spatial relationships from simple narrative text. Constraint satisfaction successfully models small worlds, though complex or ambiguous descriptions remain challenging. The maps that are rendered correctly correspond to the worlds described within the text and are visually pleasing to the eye.

5 Related Work

5.1 Narrative Understanding → Structured Representations

Early and modern NLP approaches extract who did what, where, and how from prose – via dependency parsing, SRL, and relation extraction [1] [2] [3]. These approaches then convert unstructured text into event/role graphs [4]. Prior systems show strong results on news and factual domains, and increasingly on literary text with LLMs. However, most works stop at symbolic extraction; they do not bind narrative relations to a spatial model. *Story Mapper* extends this line by mapping extracted relations (source, destination, direction, duration) into spatial constraints.

5.2 Spatial Language & Qualitative/Quantitative Reasoning

Research on spatial semantics and qualitative spatial reasoning (e.g. “north of,” “near,” “between”) provides calculi and formalisms to reason under uncertainty [5] [6], while metric approaches model distances and angles [7] [8]. These strands rarely meet narrative NLP at scale: qualitative works assume clean predicates; quantitative works assume numeric inputs. *Story Map* bridges both by normalizing qualitative claims into directional constraints and converting time/terrain/mode into approximate metric distances [9] [10], enabling a hybrid solver to place locations and detect contradictions.

5.3 Constraint Satisfaction, Inconsistency Handling, and Map Layout

Constraint Satisfaction Problems are widely used to fit models to noisy observations and to flag contradictions when constraints cannot be jointly satisfied [11]. Prior applications include scheduling, mapping, and knowledge-base repair, but rarely target fictional geographies derived from prose. *Story Mapper* formalizes narrative cues as constraints over 2D coordinates, solves them with least-squares/penalized objectives, and surfaces minimal-conflict sets as actionable feedback to authors.

5.4 Text-to-Map, Geospatial Narratives, and GIS

“Geospatial narratives” connect stories to places, often by anchoring entities to known coordinates or by visualizing paths on existing maps [12]. These systems typically require explicit geotags or post-hoc annotation. In contrast, *Story Mapper* works in invented worlds with no ground-truth lat/long, deriving the map purely from intra-textual relational cues and maintaining coherence as the manuscript evolves [9] [10].

5.5 Co-Creative Authoring Tools & Computational Creativity

Co-creative systems support writers with plot suggestions, character aids, or stylistic feedback [13] [14]. Most focus on textual assistance (summaries, continuations). *Story Mapper* contributes a complementary modality: spatial co-creation. By visualizing the world and flagging inconsistencies in situ (in the original place i.e. Google Docs), it aligns with computational creativity's goal of augmenting – not replacing – the human creator.

6 Conclusions

This project set out to explore the question: Can we leverage Natural Language Processing (NLP) and constraint satisfaction to automatically generate maps from narrative text as authors write? This was, in effect, the question of whether narrative text – especially fictional, world-building prose – could be converted into spatial constraints that generate a consistent, evolving map?

To answer this, *Story Mapper* integrates techniques from Natural Language Processing, constraint satisfaction, and computational creativity into a Google Docs plug-in that analyzes authors' writing as they work. The system extracts spatial relations (e.g. *north of*, *30 days apart*), translates them into mathematical constraints, and solves for a coherent spatial layout. This spatial layout is then presented to the author as a map.

The results demonstrate that NLP and constraint-based reasoning can successfully interpret directional and distance cues from narrative text to produce readable, dynamic maps. The tool also highlights inconsistencies – instances where the text describes impossible spatial configurations – providing actionable feedback to authors without interrupting their creative flow.

The main contribution of this work is twofold:

1. Technical: a novel pipeline that bridges unstructured narrative language and formal spatial reasoning, combining modern NLP models with traditional constraint-solving techniques.
2. Creative: a demonstration that computational systems can *augment* rather than replace human creativity by maintaining logical coherence in complex fictional worlds.

Beyond its immediate implementation, *Story Mapper* suggests broader implications for human-AI collaboration in creative domains. Future work will expand this framework to handle more abstract or metaphorical spatial descriptions, integrate learning from larger corpora, and explore three-dimensional or temporal visualizations of narrative worlds. It will also include reducing the processing time to make the generation of the map feel instant.

In sum, *Story Mapper* shows that the gap between language and space can be computationally bridged, empowering storytellers to quite literally *see* the worlds they imagine.

Bibliography

[1] Santana, B., Campos, R., Amorim, E. *et al.* A survey on narrative extraction from textual data. *Artif Intell Rev* 56, 8393–8435 (2023). https://doi.org/10.1007/s10462-022-10338-7
[2] Pawar, S., Palshikar, G. K., & Bhattacharyya, P. (2017). *Relation Extraction: A Survey.* arXiv:1712.05191 [cs.CL].
[3] Labatut, V., & Bost, X. (2019). *Extraction and Analysis of Fictional Character Networks: A Survey.* arXiv:1907.02704 [cs.CL].
[4] Chen, Y.-C. (2025). *Structured Graph Representations for Visual Narrative Reasoning: A Hierarchical Framework for Comics.* arXiv:2506.10008 [cs.CL].
[5] Cohn, A. G., & Renz, J. (2008). *Qualitative Spatial Representation and Reasoning.* In F. van Harmelen et al. (Eds.), *Handbook of Knowledge Representation* (pp. 551–596). Elsevier.
[6] Herweg, B., Koller, D., & Klippel, A. (2004). *Computing Distances and Directions for Spatial Language.* In *Spatial Cognition IV* (pp. 161–180). Springer.
[7] Freksa, C. (1991). "Qualitative spatial reasoning." In *Cognitive and Linguistic Aspects of Geographic Space*, edited by D. M. Mark & A. U. Frank, pp. 361–372. Kluwer.
[8] Andrew U. Frank (1992) — *Qualitative spatial reasoning about distances and directions in geographic space. Journal of Visual Languages & Computing*, 3(4): 343–371.
[9] Mai, G., Huang, W., Cai, L., Zhu, R., & Lao, N. (2021). *Narrative Cartography with Knowledge Graphs.*arXiv:2112.00970 [cs.GT].
[10] Mai, G., Huang, W., Cai, L., Zhu, R., & Lao, N. (2022). *Narrative Cartography with Knowledge Graphs. Journal of Geovisualization and Spatial Analysis, 6*(4).
[11] Tsang, E. (1993). *Foundations of Constraint Satisfaction.* Academic Press.
[12] Boots, B. (2004). Representations of Space and Time. *The Canadian Geographer*, *48*(2), 240+. https://link.gale.com/apps/doc/A119072956/AONE?u=anon~12e1cb59&sid=googleScholar&xid=3b9daaae
[13] Gervás, P. (2009). *Computational Approaches to Storytelling and Creativity.* AI Magazine, 30(3), 49–62.
[14] Riedl, M. O., & Young, R. M. (2010). *Narrative Planning: Balancing Plot and Character. Journal of Artificial Intelligence Research, 39*, 217–268.

A Appendix

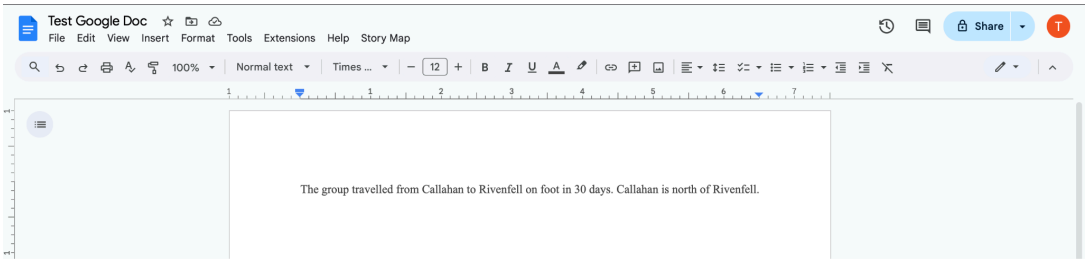

Figure A.1: The user interface featuring the *Story Mapper* Google Docs plug-in

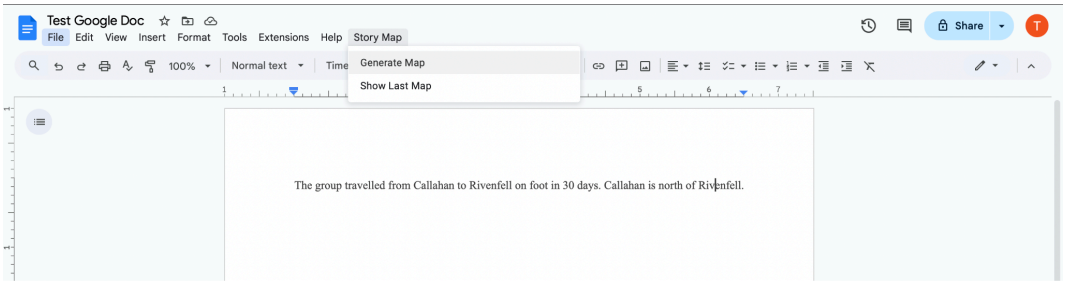

Figure A.2: The user selects the 'Story Map' menu button

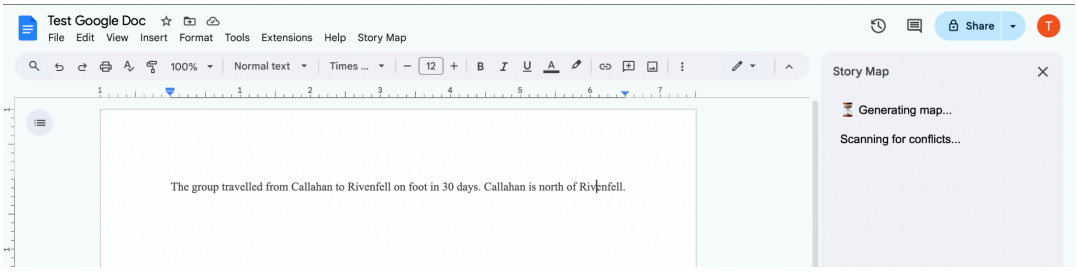

Figure A.3: The user selects 'Generate Map'

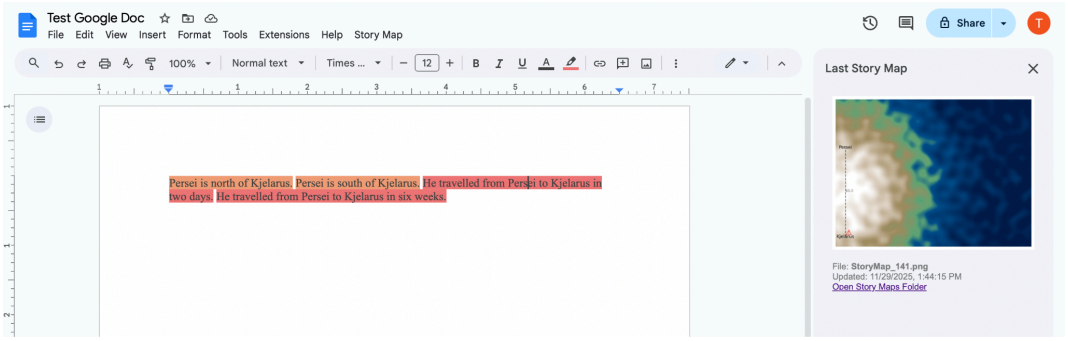

Figure A.4: The user selects 'Show Last Map'. This figures features a different input, one with conflicts.

B Appendix

Input Example 2: He went from Ayo to Lok to Mizani in a week.

In this case, *a week* corresponds to *140km* using the average travel speed of 20km per day. Unlike other examples, this *140km* is a constraint between three locations, not two. The system solves this by splitting the distance equally. Thus it becomes the case that the distance between Ayo and Lok cannot be more than 70km and the distance between Lok and Mizani can similarly not be greater than 70km if it is to be possible for the traveller to visit all three in the span of a week.

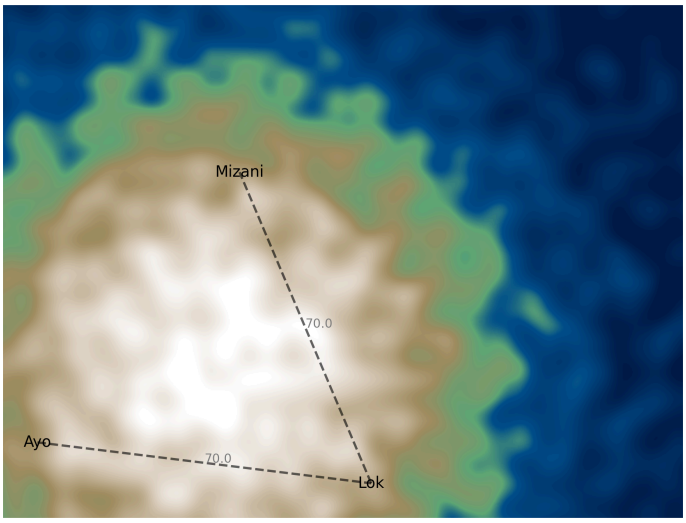

Figure B.1

Input Example 3: She went north from Troye to Kiv in one day. She went from Troye to Kiv in 30 days. Kiv is south of Troye.

In this case, we have two conflicts. The first is a distance conflict. Troye and Kiv are described as being both one day (at most 20km) apart as well as 30 days (at most 600km) apart. Kiv is also described as being both north and south of Troye.

To fully understand this example let us consider an example 3.1: “Persei is north of Kjelarus. Persei is south of Kjelarus. He travelled from Persei to Kjelarus in two days. He travelled from Persei to Kjelarus in six weeks.” In this case, there are also two conflicts but there are two distinct pairs of conflicts. In figure B.2, this can be seen visually in the highlights where each pair of conflicts is highlighted in a different color.

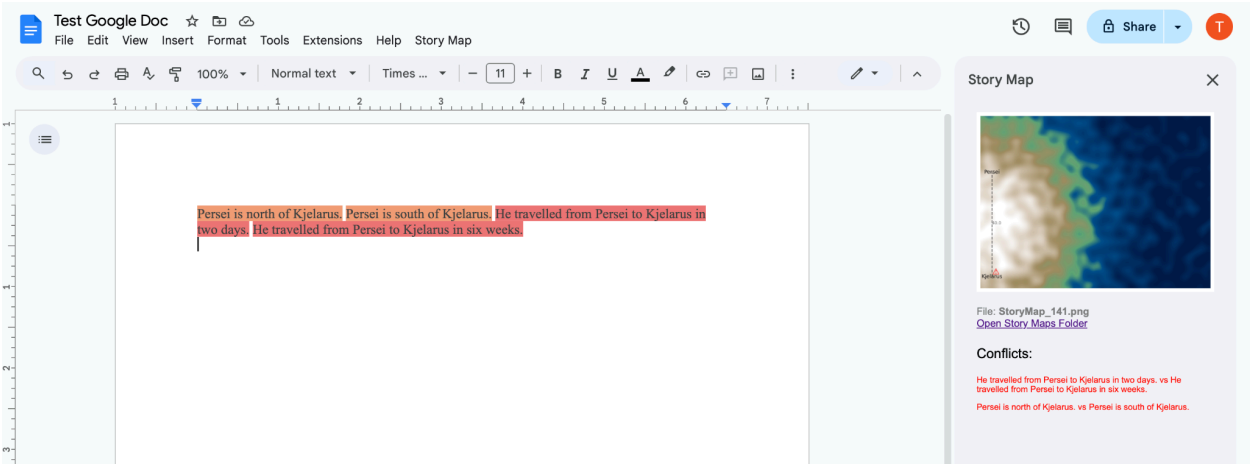

Figure B.2

In example 3, however, the two conflicts share one sentence. The system resolves this by highlighting the shared sentence in both colors as can be seen in figure B.3.

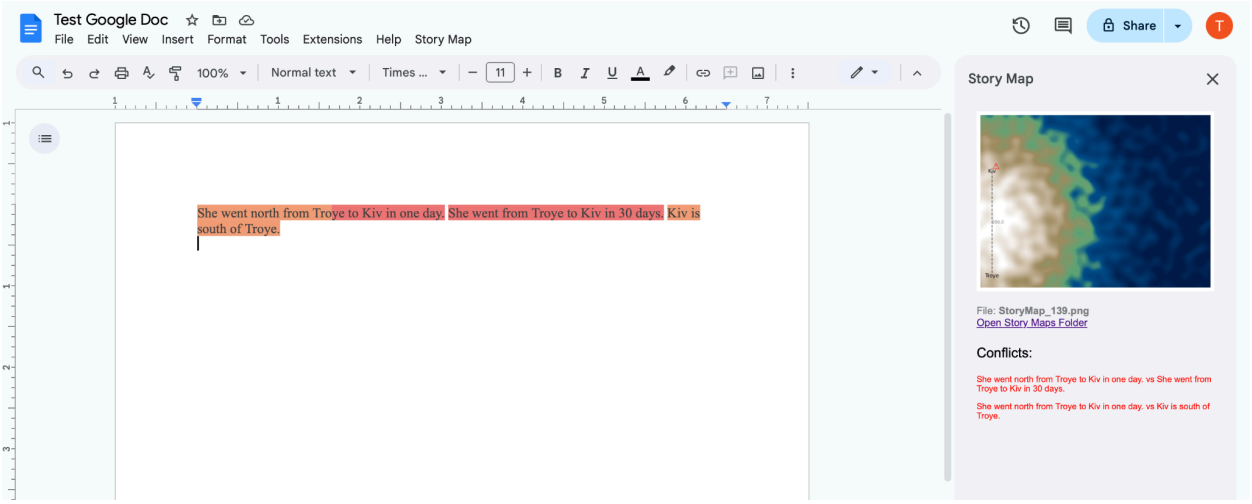

Figure B.3

Input Example 4: When her father travelled north from Rivenfell to Callahan it took him three weeks and for those three weeks Alina would turn into the most sour little girl her small town had ever seen. See, Callahan was southeast of Stonebrooke, the little trading town where Alina's mother had grown up.

This example is most similar to the inputs that the program will actually receive, where spatial information is presented alongside and within more general narrative text. Figure B.4 depicts that in this case *Story Mapper* is still able to extract and present the relevant information.

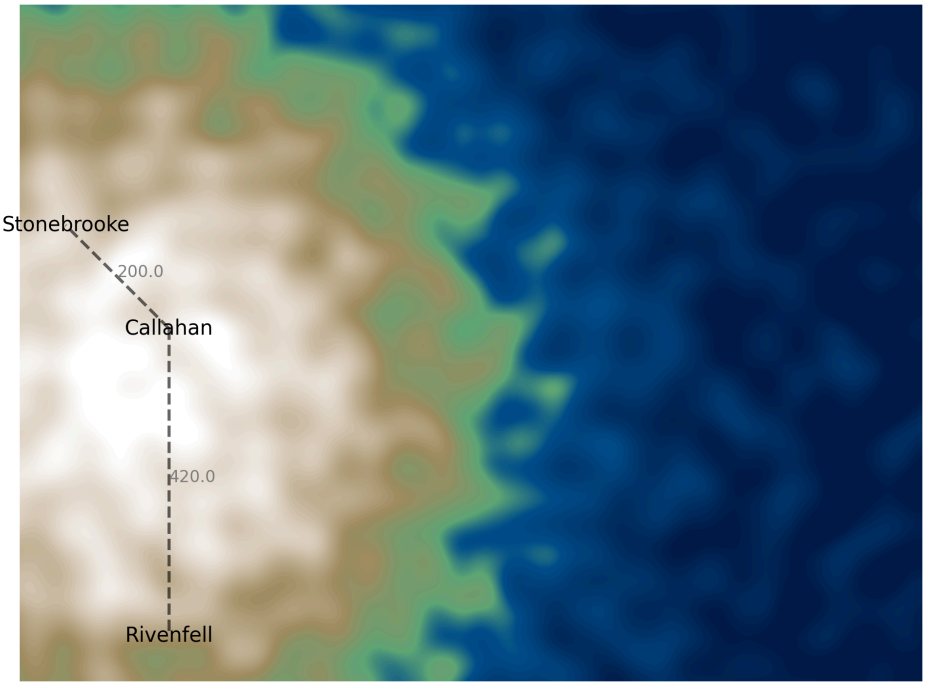

Figure B.4