Lexipedia

An Open-Source Business Process Repository

Karthika Solai, Jonathan Brandin, Alden Swain, Zachary Lisman

Abstract

In this study, we have found ways to help take the complex process of starting a business and help simplify the legal procedures. We have done this by creating process models and knowledge graphs, and through integration with Large Language Models created a database that encompasses all the connections in the codes for the State of Virginia. This is the start of a process that will have future iterations and eventually evolve into a one-stop shop wiki that will have everything that is needed to be a legally compliant LLC in whatever state you plan on operating in.

Introduction / Project Goal / Research Questions

Starting a new business in the United States can be tricky. There is lots of complicated legal jargon that makes it hard to navigate. People want to have step-by-step instructions on how the process works. Business development agencies and firms want to be able to track how new businesses are proceeding through certain milestones. The goal of the Lexipedia project is to create an open-source wiki with business process models that make complex legal processes more understandable. The models generated should be legally and ethically viable so they can be upheld in a court of law, meaning errors should be limited.

Existing literature shows how different process mapping techniques have been used to transform legal code into more digestible processes. Older research uses mapping techniques such as Petri Nets and POWL models. Newer techniques involve the use of creating knowledge graphs to implement RAG systems.

During this phase of the project, we focus on creating a scalable process that requires minimal human intervention to transform law code from any region in the United States into process models. To keep the scope tight, we've chosen to focus on local law code for places such as Charlottesville and Virginia. Through the use of LLM resources, Streamlit, and Neo4j, we focus on creating a back-end database of knowledge from law code that can then be queried to get reasonable and specific answers about business-related laws.

Literature Review / Background / Related Work

The development of our business process Wiki draws upon several areas of research, including legal data processing, process modeling techniques, and applications of large language models (LLMs) to business processes. This section reviews key literature that informs our approach.

Legal data typically contains extensively technical language that is difficult to understand for lay people. Even with LLMs being generally well-suited for natural language understanding, the "legalese" of legal documents bottlenecks the processing and understanding ability of current models. A study by Sancheti et al. (2022) describes the development of a specialized dataset, called LEXDEMOD, that provides annotated agent-specific deontic modalities. This dataset allows LLMs to be fine-tuned to better understand the nuances of legal obligations and rights,

each of which is fundamental to legal data comprehension. The LEXDEMOD corpus was shown to substantially improve transformer-based models understanding of legal contracts compared to traditional methods and showed promise for generalization capabilities but also emphasized that further work is needed to improve performance across more diverse legal documents.

There are two significant limitations found in existing process modeling notations; some notations excel at representing partial order relations, while others excel at handling control-flow constructs and hierarchy (Kourani and van Zelst, 2023). Partially Ordered Workflow Language (POWL) addresses each of these limitations by combining a partially ordered graph with control-flow operators. POWL can precisely model processes with non-hierarchical dependencies without duplicating activities, which is a common limitation of process trees. Implementation of this approach would allow us to create intuitive representations of legal procedures. Additionally, work has been done with Petri net models to aid in the understanding and controlling of business processes. Petri nets allow for a sort of formal verification method to maintain compliance while providing flexibility. Stremersch and Boel (1999) showed that Petri nets could have a "supervisory control" to ensure that the system's state always remains within a specific legal set, and Watanabe et al. (1989) showed that Petri nets can aid in the understanding of the possible execution sequences and reachability of states within a business process model. Each of these offers a method for analyzing complex interconnected processes like those in knowledge graphs.

For our comparative evaluation of LLMs on business process queries, we were interested in prior research on specialized language models for business processes. Bernardi et al. (2024) introduced the Business Process Large Language Model (BPLLM) framework that enriches process-specific knowledge. This framework couples Retrieval-augmented Generation (RAG) with fine-tuning and incorporates a process-aware chunking approach that further enhances the pipeline. The goal of BPLLM is to support users in a wide range of process comprehension and execution tasks using natural language. This study indicated that RAG-based approaches significantly outperform pure LLMs in accuracy for business process queries.

While somewhat beyond the scope of our work, current approaches to integrating AI into the cloud using centralized workflows could be problematic to the various tasks we aim to accomplish. Centralized AI workflows tend to be "walled gardens" where the flow of data between apps is limited and can hinder the ability to process information, share knowledge, and cooperate efficiently. Naptha AI (2024) proposes a protocol that focuses on modularity, decentralized data, the integration of LLMs, and flexible workflow orchestration. This would lay the groundwork for a system where combining business process modeling with knowledge graphs and LLM-based reasoning can be achieved with relative ease compared to existing centralized approaches.

While existing literature offers valuable insights into individual aspects of business process modeling and legal data understanding, there remains a gap in integrating these approaches, specifically for small businesses. Our project aims to address this gap by combining business process modeling with knowledge graphs and LLM-based reasoning to create an accessible wiki that simplifies business establishment and compliance procedures.

Data Description

Our data are law codes from all over the United States. Therefore, the formats vary drastically. They range from PDF's, CSV's, text stored on website, to XML files. Although most of our data is a variety of different formats, all of it is text based. We also assume that the language is similar. The Charlottesville dataset is all stored in XML Henrico law code in the form of PDF files. Virginia datasets come in the form of CSV's or from the Pile of Law dataset.

We processed the City of Charlottesville Municipal Code into data which could be used to generate a directional graph network from the results of the OpenAI and Mistral models. The output from the models included a source node ID, source node type, relationship between the source and target nodes, target node ID, and target node type. Each row in the dataset represented a source, relationship, target triplet. For example, one triplet from the OpenAI dataset was "Parking Meters, Installed In, Parking Meter Zone". The source node "Parking Meters" had a node type of "Equipment", and the target node "Parking Meter Zone" had a node type of "Location".

Methodology

There were several different iterations for this phase of the project. The first iteration focused on integrating systems with code and creating step by step processes. This was a two-prong process. In some instances, we manually processed laws through the code or other sites that were focused on describing these laws as process steps. In some cases, we used Dify AI to create small LLM apps powered by a limited RAG engine to automatically generate processes. The RAG engine was built using data scraped for the Open-Source Legislation Project by Will Diamond. This data was converted into smaller text files so Dify could ingest the data.

Once process steps were created, they were then modeled using the open source BPMN model maker, BPMN Assistant, created by Josip Licardo. The bpmn-assistant allows for different LLM connections. With this, we were able to test a variety of LLMs from Open AI, Qwen, and Anthropic to compare output on BPMN models.

After these initial tests were completed, we moved on to focusing on setting up a custom-built database for our needs. To do this, we have explored using python and LLMs such as OpenAI and Mistral to process written law code into Knowledge Graphs and POWL models. We used the Python package NetworkX, designed for complex network analysis, to create directional graph networks with the resulting triplets. Our goal was to perform an analysis of the graph networks generated by both LLMs, and to identify similarities and differences between them to evaluate our broader methodology of using LLMs to process legal code into a format which can ultimately be leveraged for process models or RAG. To analyze these outputs, we have created systems using NetworkX that will process valued metrics and compare the outputs and help visualize the structure. Our metrics for comparison include Betweenness Centrality, Clustering Coefficient, Degree Centrality, In-Degree Centrality, and Out-Degree Centrality.

Results and Discussion

We created 38 BPMN models covering various aspects of starting and maintaining business in Virginia and navigating the Institutional Review Board (IRB) process for human subject research. Of these models, 22 were generated using Dify AI to extract process steps. They were simplistic and did not contain enough steps for someone to create a business without needing to use other resources. 16 were created for the IRB processes by manually reading and creating the steps. These are much more thorough, but they're not scalable without significant human labor.

To make more scalable processes, we've moved to Knowledge graph. Graph networks were created by OpenAI and Mistral. The largest difference was that OpenAI identified 8.1K distinct nodes, while Mistral identified 12.9K. While this is a significant difference, it does not mean that one graph is more representative of the underlying legal code than the other. For example, the Mistral graph could be creating redundant nodes that should be consolidated.

A second key difference is that the Mistral graph has 8.9K more edges than the OpenAI graph, along with a larger number of edges per node, meaning that the Mistral model is larger and more sparsely populated than the OpenAI network.

Metric	OpenAI Graph	Mistral Graph
Node Count	8145	12862
Edge Count	12506	21401
Average Degree	1.5354	1.6639

Next, we reviewed the number of distinct node types across the networks. One potential limitation of our approach is that the graph networks we created with NetworkX do not capture node types. For example, the node "City Manager" could have a type of "Position" or "Person" depending on which triplet you are looking at. In our networks there would only be one node called "City Manager". However, it is still useful to consider the node types to get a better sense of how well the models captured nuanced legal information. Interestingly, when it came to node types, we saw opposite results compared to what we saw with node IDs. The OpenAI graph had more source and target node types than the Mistral graph. This was also true for relationship types even though the Mistral graph has far more nodes and edges than the OpenAI graph. This could indicate that the OpenAI network is encoding more of the underlying legal information in the node types and relationships, whereas Mistral is encoding more in the node IDs.

Metric	OpenAI Graph	Mistral Graph
Source Node Types	751	501
Target Node Types	1286	843
Relationship Types	4537	4139

Next, we used the Girvan-Newman algorithm to detect communities in each network. This algorithm works by detecting edges with the highest Betweenness Centrality and systematically removing those edges until distinct communities of nodes emerge. The results between the graphs were similar for the number and average size of communities. The more striking result was that each graph had a very large community at its center. The largest communities in the OpenAI and Mistral graphs had 6.7K and 11.4K nodes respectively, which both contained over

80% of all nodes in their respective graphs. This indicates that each network revolves around one very large component, with smaller communities on the fringes of the network.

Metric	OpenAI Graph	Mistral Graph
Number of Communities	408	491
Average Community Size	19.96	26.20
Max. Community Size	6726	11411

Further, we investigated the importance of key nodes, node types, and relationships in the triplet data. We found there was significant overlap in the common node types in both datasets. The types "Concept", "Person" and "Organization" were the most common. Specifically, 24% of nodes in the Mistral data were classified as concepts, while only 13% in the OpenAI data were. This could indicate that the Mistral model was failing to grasp the underlying complexity of various nodes and instead was grouping them in the very general category of "Concept". We also observed a similar pattern when looking at the frequency of relationship types. The most common relationship in both networks was "Includes". It made up about 2% of relationships in the OpenAI graph, but 4.6% in the Mistral graph. In general, the top ten Mistral relationship types were highly redundant, containing "Includes", "Has", "Part_Of", "Include", "Contains", and "Involves", all of which express very similar concepts. The most common OpenAI relationships exhibited far less redundancies. They also captured seemingly key legal concepts like "Prohibition" and "Exception" that did not appear in the top ten most common Mistral relationship types. We also looked at the most common triplets of source node type, relationship, and target node type and again found that Mistral did not appear to capture complexities in the data as well as OpenAI. For example, by far the most common Mistral triplet was "Concept Includes Concept" which was over three times as common as the next triplet. Conversely, the most common triplet in the OpenAI data was "Vehicle License_Fee Amount", which seems to capture specific and relevant legal information.

Moving beyond node type and relationships, we next inspected the key nodes in each graph. Both graphs had similar nodes which were extremely central in the network, including "City", "City Council", and "Person". The "City" node was especially vital in the OpenAI network based on metrics like degree and betweenness centrality. While it may make sense for this node to be critical given the source data was the city municipal code, our concern is that the centrality of this node could be obscuring other important structures or nodes in the graph. To understand these nodes further, we visualized the nodes in each graph based on our chosen metrics.
The first graph of Degree Centrality and Clustering (Appendix Figure 1) shows the blue City node towards the right edge of the chart with a Degree Centrality of about 0.05, far removed from the next closest node on that metric. This chart also shows that the blue OpenAI nodes tend to have higher combinations of Clustering and Degree Centrality – that is, they tend to push closer to the top right quadrant of the chart. This further emphasizes that the OpenAI network tends to have nodes that are both more central in the network and surrounded by other closely linked nodes. We also noted that the red Mistral nodes tended to cluster more in the lower left corner of the chart, which indicates that it is a sparser network.

Nest, we reviewed a scatterplot of Mistral degree balance on the Y-axis, and OpenAI degree balance on the X-axis (Appendix Figure 2). Degree balance was calculated as the In-Degree Centrality less the Out-Degree Centrality. The goal of this chart was to show us if the networks were viewing nodes differently based direction of edges. While many points did cluster around zero because nodes tended to have low values of each metric, we also saw some divergence. For example, the point for the node "Fine" has a positive degree balance in the Mistral network, but a negative balance on the OpenAI network. This could illustrate whether the LLM interprets that node as acting upon other nodes or having other nodes act upon it. It isn't clear which is correct in many cases – for example the triplets of "Person Pays Fine" and "Fine Imposed_by Judge" are both reasonable given the context - but degree balance is one metric amongst many which can help users evaluate the graph network data generated by LLMs. For our test case, the graph networks generated by both LLMs had many similarities, but the OpenAI network did appear to do a better job of avoiding redundant and generic node and relationship types, perhaps capturing more of the complexity of the underlying legal data in a meaningful way.

Conclusion / Future Work / Recommendations

The knowledge graphs we created establish the relationships within legal data. Our work lays a foundation for the next group to create BPMN models with more understanding of what these models should entail. As seen above, OpenAI outperforms Mistral in that OpenAI is less prone to duplicating nodes and relationships. This finding aligns with the study by Sancheti et al. (2022) when they demonstrated that different models have varying capabilities in detecting deontic modalities in legal texts. Further exploration could include using embedding techniques to see if the additional nodes in the Mistral graph are truly capturing new concepts not present in the OpenAI graph.

There remains much to be done before an open-source wiki can be released. The BPMN models created in the first iteration need to be reviewed by lawyers with expertise in those areas. These models can be made publicly available once they are approved. If approved, we can continue to build off them and create more that deal with more specific and targeted laws and regulations.

There could also be more testing on the effectiveness of using a Knowledge Graph versus POWL model in Neo4j for Cypher queries. Either of these could be used as part of a RAG implementation of the Gen AI stack from Docker. From there, the LLM could be used to create BPMN models surrounding business processes.

Sources

Bernardi, M. L., Casciani, A., Cimitile, M., & Marrella, A. (2024, March). *Conversing with business process-aware LargeLanguage Models: the BPLLM framework*. ResearchGate. https://www.researchgate.net/publication/379111520_Conversing_with_business_process-aware_Large_Language_Models_the_BPLLM_framework

Kourani, H., & van Zelst, S. (2023). *POWL: Partially Ordered Workflow Language*. sebastiaanvanzelst.com. https://sebastiaanvanzelst.com/wp-content/uploads/2023/08/paper_6723.pdf

Naptha AI. (2024, April 8). *Platform for Decentralized AI Workflow & Agent Orchestration*. https://www.zotero.org/groups/5968460/lexipedia/items/GPJ6G4J6/reader

Sancheti, A., Garimella, A., Srinivasan, B. V., & Rudinger, R. (2022, November 23). *Agent-Specific Deontic Modality Detection in Legal Language*. arXiv.org. https://arxiv.org/abs/2211.12752

Stremersch, G., & Boel, R. K. (1999, December). *Controlled Petri nets and general legal sets*. IEEE Xplore. https://ieeexplore.ieee.org/document/830286

Watanabe, T., Mizobata, Y., & Onaga, K. (1989, December). *Legal firing sequence and related problems of petri nets*. IEEE Xplore. https://ieeexplore.ieee.org/document/68561/similar

Appendix

Figure 1 - Degree Centrality Versus Clustering

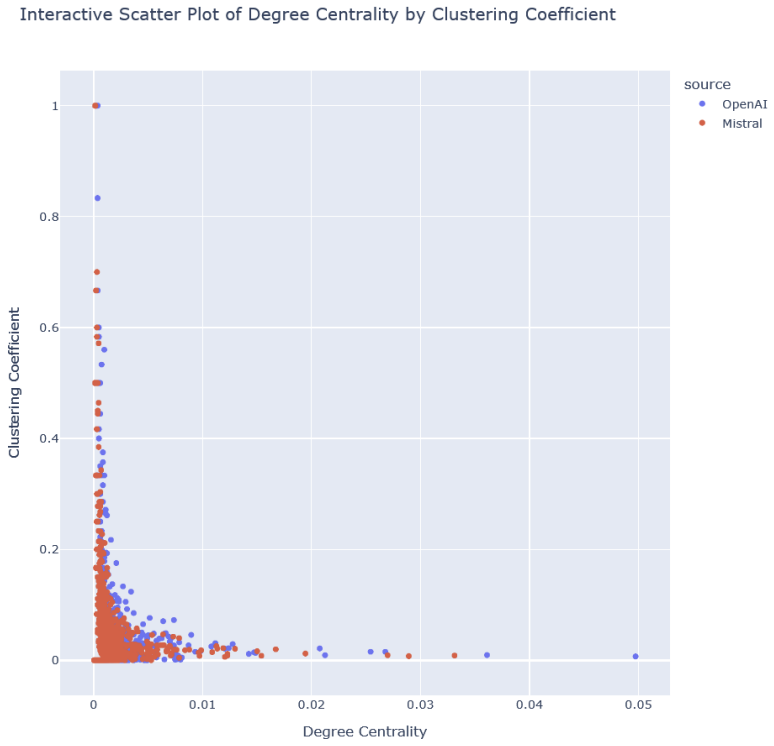
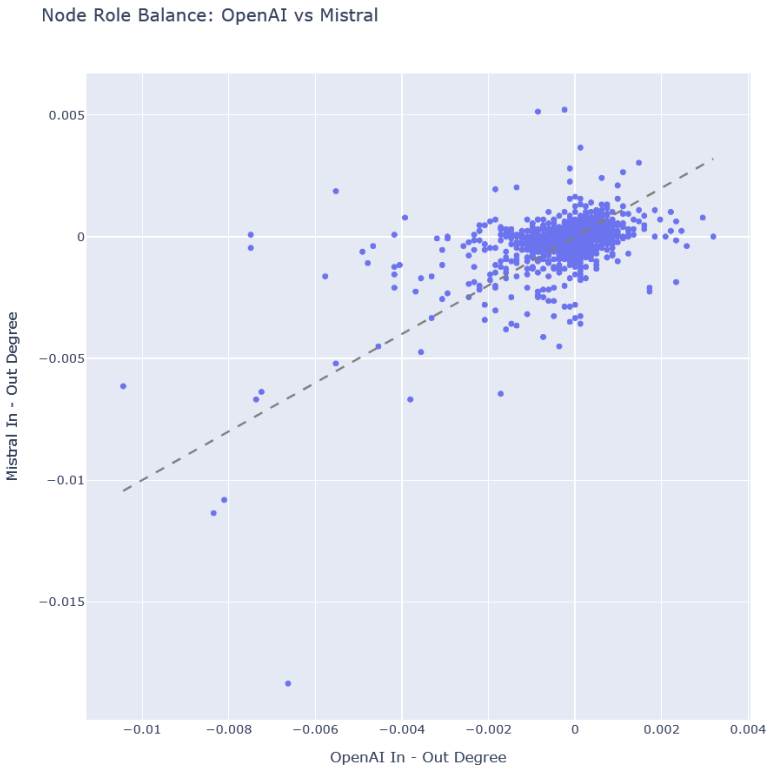


Figure 2 - Degree Balance


Node Role Balance: OpenAI vs Mistral
Mistral In - Out Degree
OpenAI In - Out Degree
0.005
0
−0.005
−0.01
−0.015
−0.01
−0.008
−0.006
−0.004
−0.002
0
0.002
0.004