Strukturierter Output für LLMs

JSON

- Strukturierter Output für LLMs wird üblicherweise als JSON File erzeugt
- Das funktioniert sowohl mit als auch ohne Schema.

JSON und OpenAI

JSON mode

JSON mode is a more basic version of the Structured Outputs feature. While JSON mode ensures that model output is valid JSON, Structured Outputs reliably matches the model's output to the schema you specify. We recommend you use Structured Outputs if it is supported for your use case.

When JSON mode is turned on, the model's output is ensured to be valid JSON, except for in some edge cases that you should detect and handle appropriately.

To turn on JSON mode with the Chat Completions or Assistants API you can set the `response_format` to `{ "type": "json_object" }`. If you are using function calling, JSON mode is always turned on.

JSON und Langchain

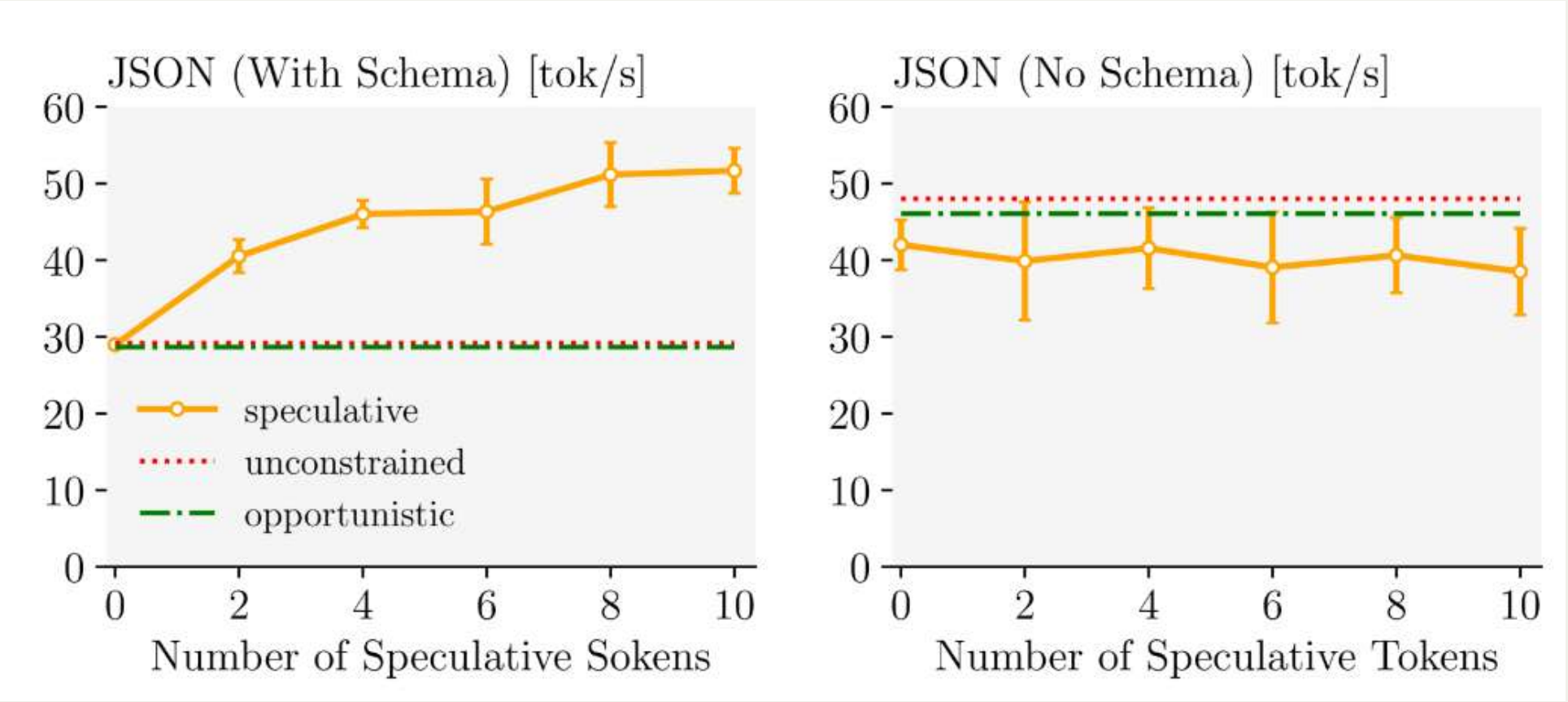
```
from langchain_core.output_parsers import JsonOutputParser
# Define your desired data structure.
class Joke(BaseModel):
    setup: str = Field(description="question to set up a joke")
    punchline: str = Field(description="answer to resolve the jok

parser = JsonOutputParser(pydantic_object=Joke)
```

Nachteile im strukturierten Output

1. Gerade mit Schema nimmt die Anzahl spekulativer Tokens im Laufe des Generationsprozessen deutlich zu.
2. Nicht alle maschinenlesbaren Sprachen eignen sich für eine Beschreibung in JSON.

Beurer-Kellner, Fischer, Vechev: Guiding LLMs the right way



BNF /GBNF

- Die Baccus-Naur-Form ist eine Standardbeschreibungsprache für alle Maschinensprachen.
- Sie wurde für Tensormodelle als GBNF-Grammar umgesetzt.
- GBNF-Grammar kann für alle Tensor Modelle zur Erzeugung eines strukturellen Outputs verwendet werden.

Das LLM darf nur mit Ja/Nein antworten:

```
root:== ("Ja"|"Nein")
```

Das LLM darf nur mit Nummern oder einem generischen Satz antworten:

```
root:== number | "Doesn't look like anything to me."
number:== [0-9]+
```

Beispiel musikalische Annotation (ABC-Notation):

```
X:1
T:The Legacy Jig
M:6/8
L:1/8
R:jig
K:G
GFG BAB | gfg gab | GFG BAB | d2A AFD |
GFG BAB | gfg gab | age edB |1 dBA AFD :|2 dBA ABd |:
efe edB | dBA ABd | efe edB | gdB ABd |
efe edB | d2d def | gfe edB |1 dBA ABd :|2 dBA AFD |]
```

file (`X:`), the title (`T:`), the time signature (`M:`), the default note length (`L:`), the type of tune (`R:`) and the key (`K:`)

Das LLM darf nur mit einer musikalischen Notation antworten:

```
root ::= tune+
tune ::= header body
header ::= field-line+
field-line ::= field-id ":" field-value newline
field-id ::= "X" | "T" | "C" | "M" | "L" | "K" | "Q" | "Z" | "N"
field-value ::= text
body ::= music-line+
music-line ::= note-group bar-line
note-group ::= note | chord | rest | decorator
note ::= pitch length
pitch ::= natural | sharp | flat
natural ::= "A" | "B" | "C" | "D" | "E" | "F" | "G"
sharp ::= natural "^"
flat ::= natural "_"
length ::= fraction
```

GBNF und Argument Mapping

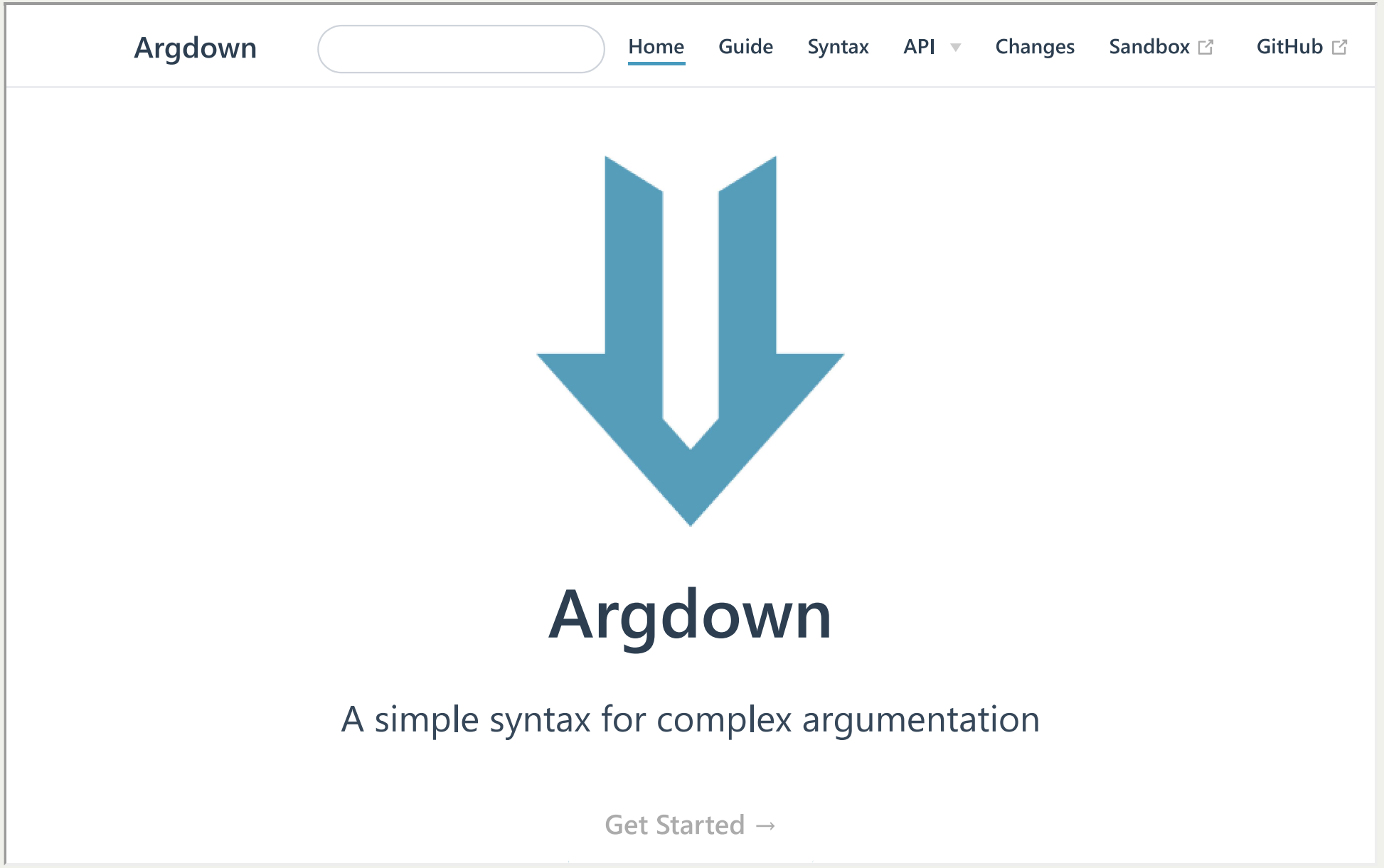
Als Argument Mapping bezeichnet man die primär anaolge Erstellung von Argument Maps bzw. Refutation Maps bei der Argumente und Ihre Beziehung zueinander als Karte erstellt werden.

Argdown

Was ist Argdown?

Die auf Markdown basierende Auszeichnungsprache Argdown kann dazu verwendet werden um Argumente digital so zu beschreiben, dass aus Ihnen automatisch eine Argument-Map erstellt wird. Diese Visualisierung basiert auf Mosaik und kann als JSON, SVG und als HTML Web-Komponente verwendet werden.

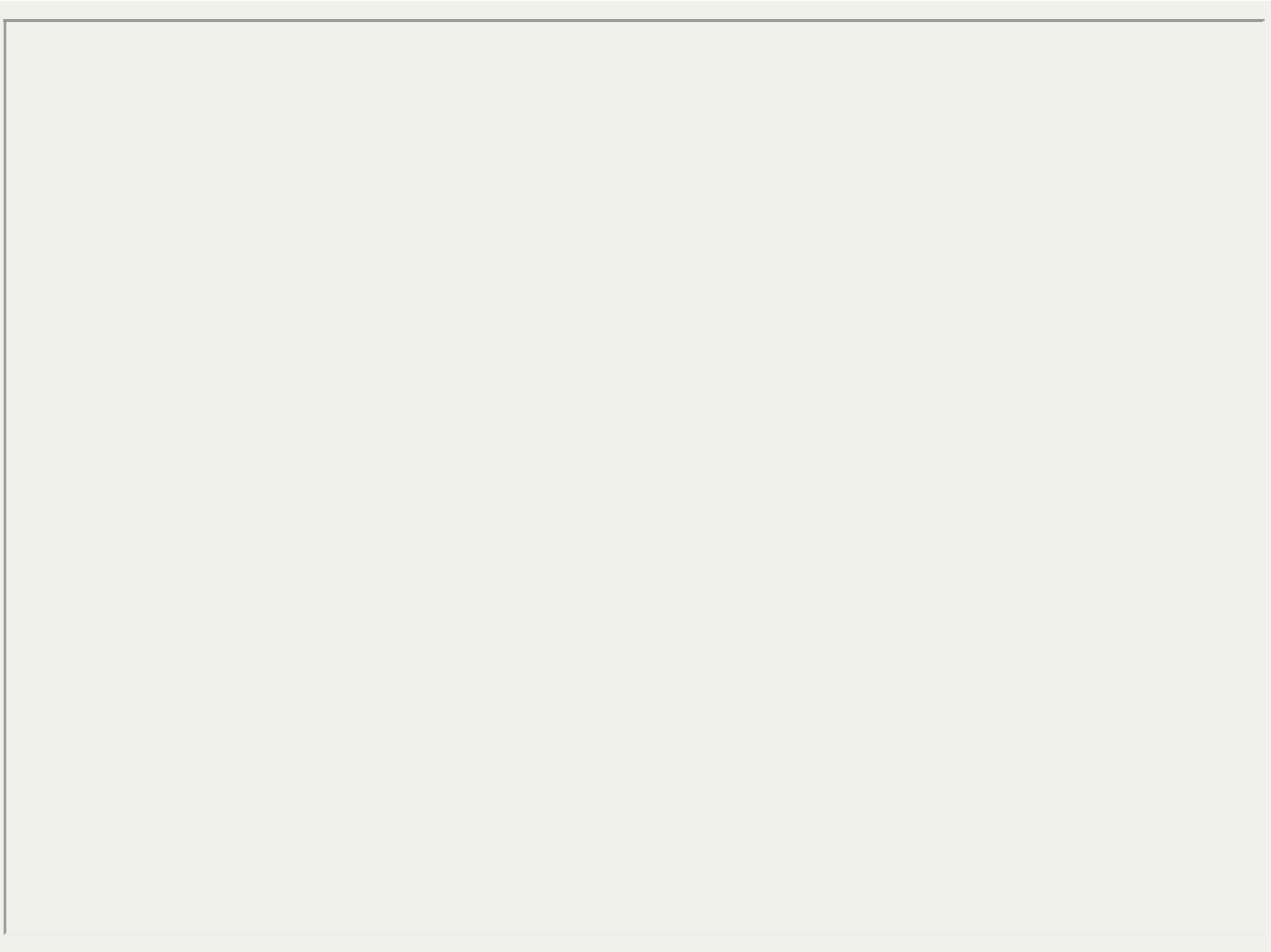
Was kann Argdown?



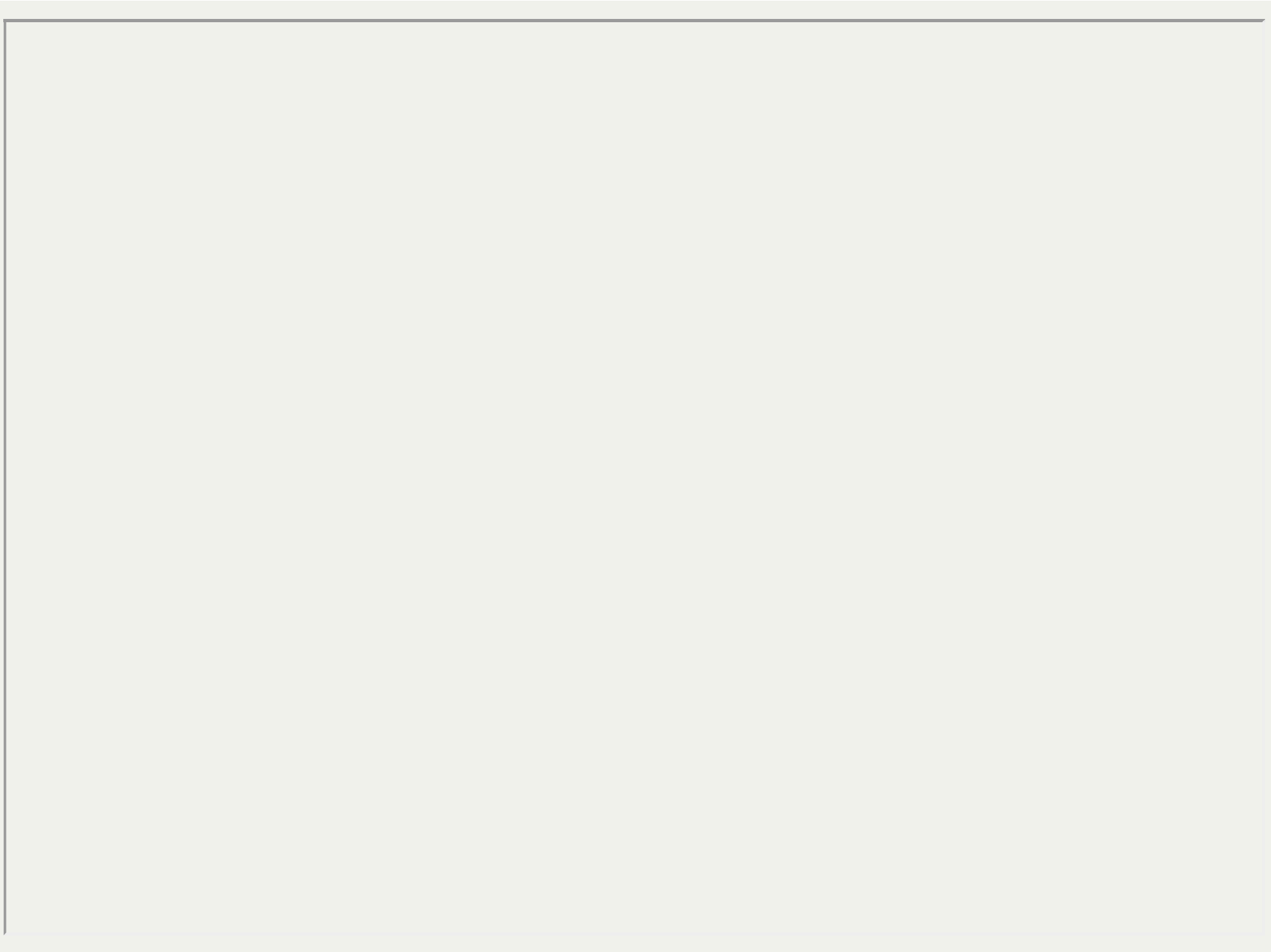
Mit GBNF kann der Argdown Syntax so beschrieben werden, dass ein LLM zuverlässig den Syntax erzeugt und damit eine Applikation erstellt werden kann.

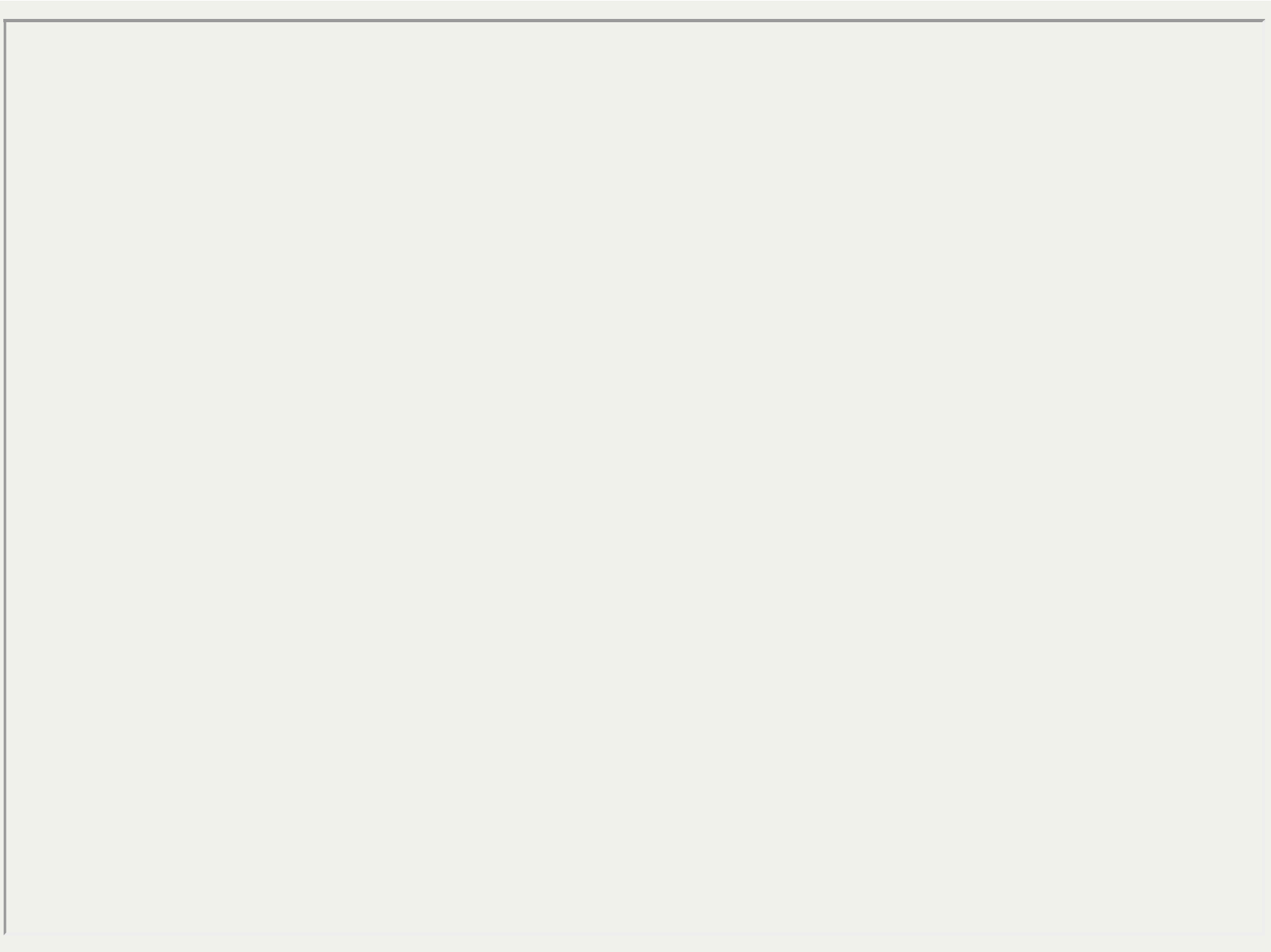
Datenbasis

Als Datenbasis dient das Regesta Imperii Projekt aus der Mediävistischen Grundlagenforschung. Die RI enthält die Zusammenfassung von mehr als 145000 Urkunden und historiographischen Quellen des Mittelalters von ca. 500 - 1500. Das Projekt hat dieses Jahr das 200ste Jubiläum.



In den Regesten finden sich rechtliche Argumente zur Ereignis- und Rechtsgeschichte. In den Kommentaren zu den Regesten die durch Regestenmitarbeiter erstellt wurden finden sich Argumente zu Forschungsfragen wie zum Beispiel Datierung und Prosopographie.





Technischer Aufbau (nicht Workflow)

- Retrieval Augmented Generation auf Basis der csv-Tabellen des RI-Repositories.
- Streamlit App für User Input und Parameter Eingabe.
- Auf Basis der Frage oder des Statements des Users werden relevante Regesten gesucht.

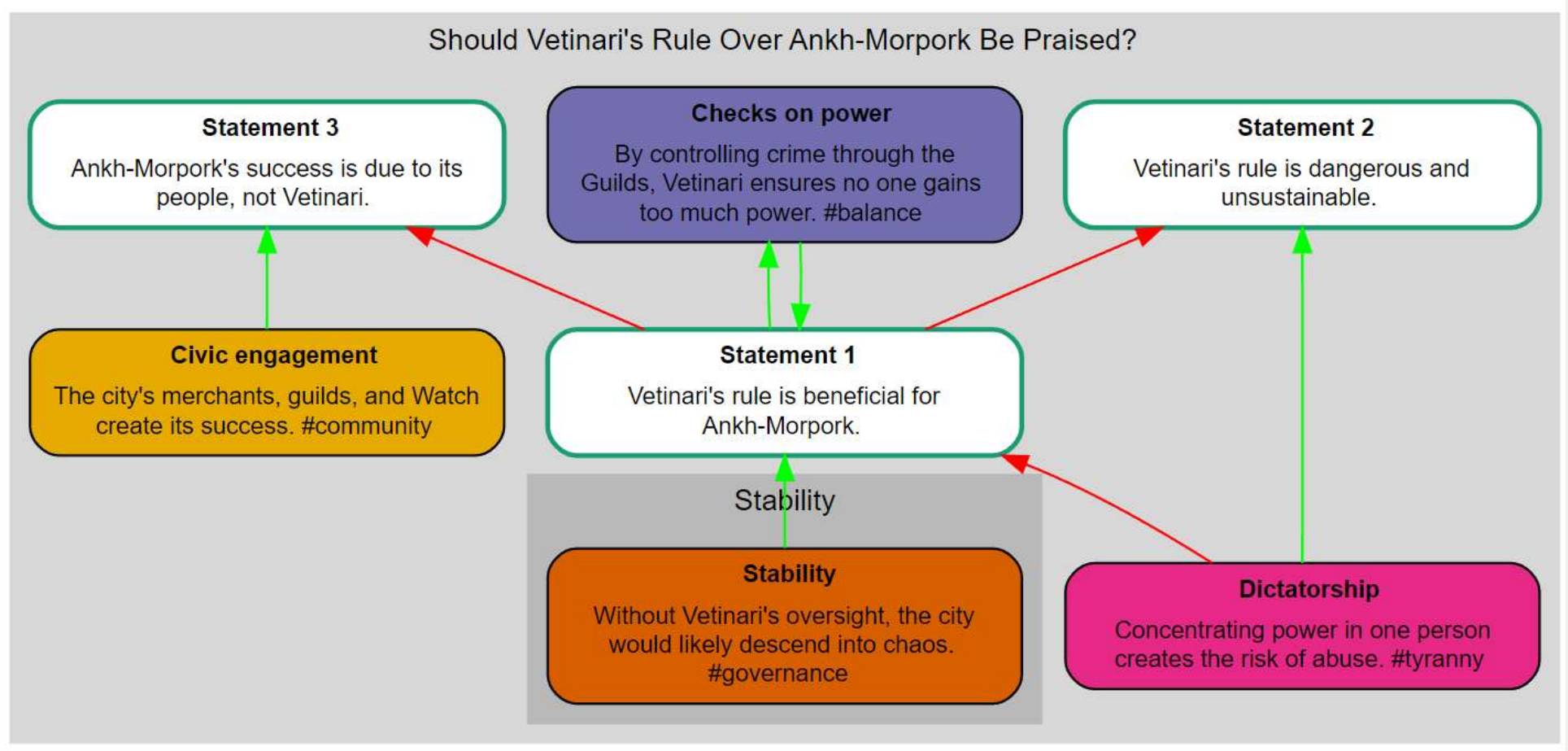
Erstellung von Dokumenten:

```
# Function to load CSV file and return documents
def createDocTypeC(file_path):
    documents = []
    with open(file_path, newline='', encoding='utf-8') as csvfile
        reader = csv.reader(csvfile, delimiter='\t')
        header = next(reader)
        for row in reader:
            content = str(row[4])
            metadata = {
                "Identifier": str(row[0]),
                "URI": "http://www.regesta-imperii.de/id/" + str(
            }
            documents.append(Document(page_content=content, metad
    return documents
```

- In diesen Regesten werden Argumentationen gesucht und über das LLM und das GBNF grammar ein strukturierter Output erzeugt.
- Der abgespeicherte Output wird an das Argdown command-Line Tool weitergegeben und so ein Web-Component erzeugt der in Streamlit dargestellt wird.

- Dabei entsteht Argument Map mit Links die Dank der Retrieval Augmented Generation mit den entsprechenden Regesten auf der Projektseite verknüpft sind.
- Den KI-Agenten wurden Beispiele von Regesten und erwünschtem Output zur Verfügung gestellt.

Eine erzeugte Argument Map:



Erzeugt mit dieser Argdown Auzeichnung:

```
# Should Vetinari's Rule Over Ankh-Morpork Be Praised?

[Intro]: In this debate, we explore whether Lord Vetinari, the Pa
## The Arguments
[Statement 1]: Vetinari's rule is beneficial for Ankh-Morpork.
  + <Stability>: Vetinari has brought unprecedented stability to
  + <Checks on power>: By controlling crime through the Guilds, V
  - <Dictatorship>: His rule is not democratic and concentrates p

[Statement 2]: Vetinari's rule is dangerous and unsustainable.
  + <Dictatorship>: Concentrating power in one person creates the
  - <Stability>: Vetinari's pragmatism ensures the city runs smoo

[Statement 3]: Ankh-Morpork's success is due to its people, not V
  + <Civic engagement>: The city's merchants, guilds, and Watch c
```

Argdown als BNF Grammar ausführlich aber daher ungeeignet für unsere Zwecke.

```
<argdown> ::= (<element> <newline>*)*

<element> ::= <heading>
            | <statement>
            | <argument>
            | <relation>
            | <list>
            | <comment>
            | <link>
            | <tag>
            | <inline-formatting>

<heading> ::= "#" <text> <newline>
            | "##" <text> <newline>
            | "###" <text> <newline>
```

Jede Einschränkung des Grammars erhöht den spekulativen Teil des Outputs. Daher sollte das Grammar möglichst kompakt sein:

```
root ::= (argument "\n" | statement "\n")+ (contradiction|confirm
statement ::= "\[" summary "\]" "\n"

agumentShort ::= "<" title ">" "\n"

argument ::= "<" title ">" "\n" "\n" "(1.) " premiss "\n" ("(2.)

title ::= [a-zA-Z_.,;-!:() ]+ "\n"

summary ::= [a-zA-Z_.,;-!:() ]+ "\n"

premiss ::= [a-zA-Z_]+ "\n"

conclusion ::= "(4.) " (statement)? [a-zA-Z_.,;-!:() ]+ "\n"
```

Einen Schritt weiter

Man kann auch ein Grammar automatisch generieren.

1. Nur Argumentationen mit GBNF generieren
2. Aus den Titeln der Argumentationen ein Grammar für den "head" in dem die Relationen ausgezeichnet werden generieren.
3. Danach beides zu einer Datei zusammenfügen.

Das erste Grammar:

```
root ::= ("<" title ">" "\n" "\n" "(1.) " premiss "\n" ("(2.) " p
modusponens ::= [a-zA-Z_.,;-!:() ]+ "\n" #Logisch_gueltige_Schlus
title ::= [a-zA-Z_.,;-!:() ]+ "\n" #kurzer_Titel_des_Arguments_fu
premiss ::= [a-zA-Z_]+ "\n" #Praemisse_des_Arguments
```

Das zweite Grammar:

```
root ::= (contradiction|confirmation)+
contradiction ::= (statement|argument) "\n\t" "-" (statement|argu
confirmation ::= (statement|argument) "\n\t" "+" (statement|argum
argument ::= {v1|v2|v3|v4[...]} #Hier_werden_die_Argumenttitel_de
```

Probleme:

1. Relevanzproblem
2. Bias
3. Abstraktionsebene
4. Wissenslücken

Relevanzproblem

1. Es muss festgelegt werden wie viele Dokumente gefunden werden sollen.
2. Daher werden in fast allen Fällen entweder zu viele oder zu wenige Dokumente gefunden.
3. Das kann man durch ein Rescaling lösen und den User miteinbeziehen.

Relevanzproblem - Lösungsansatz

1. Den User bestimmten lassen, wie viele Dokumente verwendet werden sollen.
2. Erstmal die vom User bestimmte Anzal n x 10 mit Similarity Search finden.
3. Danach mit einem Tensor Model die aus diesem Dokumenten die wichtigsten filtern.

Blog

aRIguments

Geben Sie Ihre Frage zum Inhalte der Regesta Imperii ein oder ein Statement zu dem Sie Argumentationen finden wollen:

Friedrich III. war mit allen Bischöfen befreundet.

Generieren einer Argument Map

[Document(metadata={'Identifier': 'Chmel n. 8109', 'URI': 'http://www.regesta-imperii.de/id/1487-07-18_3_0_13_0_0_8108_8109'}, page_content='Das folgende Regest enthält entweder die Zusammenfassung einer mittelalterlichen Urkunde oder eine historiographische Quelle:bevollmächtigt die Churfürsten Friedrich zu Sachsen und Johann zu Brandenburg, denselben Anschlag von den Bischöfen, Prälaten und Städten in ihren Fürstenthümern einzubringen.'), Document(metadata={'Identifier': 'Chmel n. 8109', 'URI': 'http://www.regesta-imperii.de/id/1487-07-18_3_0_13_0_0_8108_8109'}, page_content='Das folgende Regest enthält entweder die Zusammenfassung einer mittelalterlichen Urkunde oder eine historiographische Quelle:bevollmächtigt die Churfürsten Friedrich zu Sachsen und Johann zu Brandenburg, denselben Anschlag von den Bischöfen, Prälaten und Städten in ihren Fürstenthümern einzubringen.'), Document(metadata={'Identifier': 'Chmel n. 8629',

Abstraktionsebene

Individuelle Informationswerte zu Ereignissen, Orten und Personen zu finden und einzuordnen ist eine völlig andere Aufgabe. Ein LLM unterscheidet von sich aus nicht zwischen unterschiedlichen Abstraktionsebenen.

Möglicher Lösungansatz: Der Benutzer muss festlegen auf welcher Ebene das Interesse liegt und das System greift dann auf entsprechende Agenten zu die unterschiedliche Beispiele erhalten haben.

Danke für Ihre Aufmerksamkeit

Ressourcen:

https://pad.gwdg.de/RnJRU5w_TcSVFcXffg3dbg